DOCUMENT RESUME

ED 254 619 CE 040 122

AUTHOR	Matthews, John; Winsauer, John
TITLE	Computer Software for Vocational Education: Development and Evaluation. State-of-the-Art Paper.
INSTITUTION	Tennessee Univ., Knoxville. Office for Research in High Technology Education.
SPONS AGENCY	Office of Vocational and Adult Education (ED), Washington, DC.
PUB DATE	Dec 84
CONTRACT	300-83-0176
NOTE	87p.; For related documents, see CE 040 115-126. Product of the "High Technology Education: A Program of Work" Project.
PUB TYPE	Information Analyses (070)
EDRS PRICE	MF01/PC04 Plus Postage.
DESCRIPTORS	Adult Vocational Education; *Computer Assisted Instruction; *Courseware; *Evaluation Criteria; *Instructional Material Evaluation; Learning Theories; *Material Development; Measurement Techniques; Postsecondary Education; Secondary Education; State of the Art Reviews; *Vocational Education

ABSTRACT

This paper presents a basis for judging the merits and shortcomings of computer software for vocational education and contains a comprehensive evaluation tool for educational software. The historical development of computer hardware and software as used in education is addressed. A discussion of the evaluation of educational software stresses the need for outside evaluations and describes some aids to developing and evaluating software. Available evaluations are identified. Concepts from learning theory are translated to the world of educational software; these include: classical and operant conditioning, cognitive approaches, and learning efficiency and effectiveness. Instructional models that might be derived from learning theory are analyzed. Instructional methodology for computer delivery is discussed in terms of the nature of the hardware available and the appropriateness of the software to the instructional objectives--both of which place constraints on using computers for instruction. Criteria used in available instruments for evaluating educational software are presented, including software objectives and design, acquisition, program content, hardware/software operation and the user, and compatibility and special system needs. The final section describes the evaluation instrument, which was evolved from other evaluatory systems and was pilot-tested with a variety of vocationally oriented software packages. Appendixes contain the Educational Microcomputer Software Description and Evaluation Form and sample evaluations. (YLB)

Computer Software for Vocational Education: Development and Evaluation

STATE-OF-THE-ART PAPERS

OFFICE FOR RESEARCH IN HIGH TECHNOLOGY EDUCATION
The University of Tennessee College of Education

Computer Software for Vocational Education: Development and Evaluation

by

John Matthews, Professor and Head, and
John Winsauer, Graduate Research Assistant
Department of Technological and Adult Education
University of Tennessee, Knoxville

Office for Research in High Technology Education
428 Claxton Addition, College of Education
The University of Tennessee, Knoxville, TN 37996-3400

December, 1984

Sponsoring Agency:
U.S. Department of Education
Office of Vocational and Adult Education

STATE-OF-THE-ART PAPER

COMPUTER SOFTWARE FOR VOCATIONAL EDUCATION: DEVELOPMENT AND EVALUATION

PAN R01-1565-44-012-85

FUNDING INFORMATION

Project Title:	High Technology Education: A Program of Work
Contract Number:	300830176
Source of Contract:	U.S. Department of Education Office of Vocational and Adult Education
Project Monitor:	Richard DiCola
Contractor:	The University of Tennessee
Project Director:	Janet Treichel Sheila McCullough
Principal Investigators:	At Home in the Office Study - Sheila McCullough COMTASK Database - John Peterson State-of-the-Art Papers - Lillian Clinard
Disclaimer:	The activity which is the subject of this report was supported in whole or in part by the U.S. Department of Education. However, the opinions expressed herein do not necessarily reflect the position or po'icy of the Department of Education, and no official endorsement by the Department of Education should be inferred.
Discrimination Prohibited:	No person in the United States shall, on the grounds of race, color, or national origin, be excluded from participation in, be denied the benefits of, or be subjected to discrimination under any program or activity receiving Federal financial assistance, or be so treated on the basis of sex under most education programs or activities receiving Federal assistance.

FOREWORD

The Office for Research in High Technology Education at the University of Tennessee, Knoxville, is conducting a program of work on high technology and its implications for education. Funded by the U.S. Department of Education's Office of Vocational and Adult Education, the program addresses the skill requirements and social implications of a technology-oriented society. Issues concerning computer literacy and computer applications are a focus of the program. The balance between the liberal arts and technological skills and the complementary roles they play in enabling people to function in and derive satisfaction from today's high-technology era are also addressed. The program's efforts are targeted at secondary schools, two-year post-secondary institutions, community colleges, universities, industrial training personnel, and other education and training groups.

The program consists of three major components:

At Home In the Office Study - At Home In the Office is an experiment that has placed office workers and equipment in the workers' homes to determine (1) what types of office work can effectively be done at home and (2) the advantages and disadvantages of home work stations. The implications for educators, employers, and employees will be significant, as work at home offers a possible avenue of employment for people living in rural areas, parents of pre-school children, handicapped individuals, and others.

COMTASK Database - COMTASK is a model of a computerized task inventory for high-technology occupations. The outcomes of the COMTASK system include a sampling of task analyses, the demonstration of how these task analyses can be rapidly updated, a manual for conducting task analyses to provide data for the system, and a guide to using the system.

State-of-the-Art Papers - A series of nine papers is being developed to address high technology and economic issues that are of major concern to education. Nine working titles have been selected:

- The Changing Business Environment: Implications for Vocational Curricula
- Computer Literacy in Vocational Education: Perspectives and Directions
- Computer Software for Vocational Education: Development and Evaluation
- Educating for the Future: The Effects of Some Recent Legislation on Secondary Vocational Education
- The Electronic Cottage
- High Technology in Rural Settings
- (Re)Training Adults for New Office and Business Technologies
- Robots, Jobs, and Education
- Work in a World of High Technology: Problems and Prospects for Disadvantaged Workers

Abstract

In vocational education, the impact of the microcomputer as an instructional vehicle is a function of (a) the perceived instructional need, (b) the availability of high-quality software which meets that need, (c) possession of hardware capable of running the software, (d) a benefit-cost ratio that is high compared with those of other types of instruction, and (e) teachers skilled in preparing or using the software.

This paper addresses the historical development of computer hardware and software as used in education. It stresses that the development and evaluation of educational software should be a team effort based on accepted learning theory and instructional strategies. The paper includes a comprehensive evaluation instrument which was evolved from other evaluatory systems and was pilot-tested with a variety of vocationally oriented software packages.

About the Authors

John Matthews has many years of experience in the teaching of electronics at the high school and college level and has developed a computer laboratory for instruction in vocational education. He has developed criteria for the selection of computer hardware for educational applications and has used and developed numerous educational software packages.

John Winsauer is a business administration graduate studying microcomputer technology in the Department of Technological and Adult Education. He currently assists students using the computer laboratory.

About the Editors

This paper has been prepared as part of a series of state-of-the-art papers edited by Lillian A. Clinard, an associate director of The University of Tennessee's Energy, Environment, and Resources Center (EERC), and Mary R. English, a research associate at EERC. The editors, who have been on assignment to the Office for Research in High Technology Education, were responsible for selecting the series' authors, reviewing and coordinating external reviews of the papers, and preparing the papers for release.

Acknowledgments

The editors express their appreciation to Donald Holznagel, Roger Lambert, and John Peterson, who reviewed earlier drafts of the paper, and to the U.S. Department of Education's Office of Vocational and Adult Education, which also provided a review.

CONTENTS

CONTENTS (Continued)

INTRODUCTION

In business and industry, where vocational education and training is often crucial, learning can be individualized through the use of microcomputers and instructional software. For example, the automobile industry, in dealing with its thousands of agency service departments, can use educational software with graphics to quickly lead mechanics through new training. This software can be rerun until mastery is achieved; it can relate easily to prior instruction; it can come in short modules or clusters of modules integrated with video presentation. Such examples of computer software uses in industrial training have obvious links with classrooms, laboratories, and shops where vocational education is taking place. Similarly, principles of computer use that apply to vocational education also apply to other educational areas.

As a tool for aiding instruction, the microcomputer thus holds great promise for education, and especially for vocational education. The design of good educational software, however, is a major obstacle to realizing this promise. Because developing high-quality software is a complex and difficult process, virtually every field wishing to use computers must also be able to evaluate which software to use.

The Charge

This state-of-the-art paper has as its charge the following:

> This paper will set forth a basis for judging the merits and shortcomings of the computer software for vocational education. Included will be evaluation tools to enable readers to make their own critical judgments of the various vocational education software

> they encounter. In addition, it will provide an annotated review of software currently in use by vocational educators by utilizing the evaluation tools. It will not, however, attempt to give an exhaustive list of all vocational software, but will form the basis for an on-line data base of vocational education software.

As this suggests, the paper will attempt to serve two purposes: (a) to set forth the types of questions that ought to be asked about software being considered for purchase, and (b) to provide an evaluation system based on the considered judgments of a variety of software evaluators representing substantive evaluative efforts in the field. The first purpose is to make people aware of pertinent considerations in evaluating software; the second, to provide an instrument for software evaluators to use as a basis for consistent evaluation.

It should also be noted at the outset that the software evaluation instrument which accompanies this paper is complicated and may not be appropriate for novice software users. The resultant reviews, however, are quite usable by anyone who can use the software and its microcomputer. As the charge indicates, this could form the basis for an on-line data base of software reviews. (Another such data base -- Resources In Computer Education [RICE] -- now exists and may be accessed through the Bibliographic Retrieval Services [BRS] network.)

FRAME OF REFERENCE

Before anyone can fully appreciate educational software, a frame of reference must be established. What appears to be an exciting tool to advance instructional delivery is, beneath the surface, a complex system of software-hardware interaction. For example, it could take months to prepare a good educational software package that is then used to teach a set of concepts in three hours. The package must contain both the computer program, generally on a 5-1/4 inch diskette, and a binder with use instructions and directions, called documentation, that lead the user through the necessary steps. Production of a successful software package thus may entail the involvement of several people over a significant period of time. To understand and evaluate this final product, some background is helpful.

Historical Perspective: Genesis, Proliferation, and Incompatibility

Until the advent of three relatively inexpensive microcomputers (the Commodore Pet, the TRS-80 Model I, and the Apple) late in the 1970s, few in the educational community had much experience with computers. A few schools had access to timeshare computer terminals, but the annual cost per terminal was prohibitively high. However, several significant efforts with mainframe computers led the way in developing instructional software. Notable in this field, although not without problems, was the PLATO system originating with the University of Illinois in 1960 and using a Control Data Corporation mainframe computer. Perhaps somewhat more successful, though perhaps less well known, was the Timeshared Interactive Computer Controlled Information Television (TICCIT) system developed by the MITRE Corporation in 1971 in Austin, Texas, and later installed at Brigham Young University. It used a Sony color television set for both video terminal and digitized messages for audio-visual display.

As one could expect, the rapid expansion of the computer industry led to a variety of incompatible software products as the industry evolved its technology and hardware design. Delaware's experience during the 1970s provides a good example of this.

In 1972, the Delaware School Auxiliary Association (DSAA) supplied a Digital Equipment Corporation timesharing minicomputer with printing terminals for several schools, at a token annual charge of about $3,000 per terminal for hardware and software support. In time, this system, known as Project Delta, became a part of the Department of Occupational Education in the University of Delaware's College of Education. Faculty members in that department provided workshops and classes, often funded by DSAA, to prepare vocational teachers to program educational software using the Basic-Plus language. A computer-managed instruction (CMI) system was developed by the department faculty, and instructional software developed by teachers for this system was made available on both the Delta and the University computer systems.

At the same time that Project Delta was maturing, the University installed a PLATO system lab in another part of the College of Education. The main thrust of this lab was to provide computer-assisted instruction (CAI) to university classes.

In 1976, the Department of Occupational Education, while operating Project Delta, purchased one of the first microcomputer systems on the market, seeing this as the educational computer of the future. It was primitive and no software was available, but the Project Delta staff developed it into a timeshare system by 1977. They even wrote software to make the earliest version of the Control Program for Microcomputers (CP/M) operating system work in conjunction with the Technical Designs Lab's floppy-disk operating system (FDOS).

At the same time these activities were going on, Delaware installed a Hewlett-Packard Model 2000 minicomputer to deliver CAI drill and practice to many schools in the state. This system, together with the other three, were all in place and operational in early 1977, vividly demonstrating that even in one area, four separate systems were under way for the development and delivery of educational computing.

At this time, in the late 1970s, Commodore Industries introduced its PET microcomputer, followed by the Radio Shack TRS-80 and the Apple microcomputers. All these systems were almost totally incompatible, yet all were delivering needed services. Indications were that there would soon be a storm of demand for microcomputer software.

In 1979, one could subscribe to all the microcomputer journals and read them in leisure time, easily keeping up with every significant hardware and software development. By 1983, over one hundred manufacturers were making microcomputers aimed at the business and educational markets. Software programs for them were being developed by the thousands. There was a clamor of demand for compatibility. Digital Research Incorporated

(DRI) and Microsoft Corporation (MC) came to lead the field as providers of compatible operating systems, compatible languages, and integrated software systems.

While the CP/M operating system dominated the microcomputing field until 1982, the advent of the 16-bit IBM Personal Computer (IBM-PC) brought the first real rush to become compatible with some standard. By 1984, the popular 16-bit Microsoft disk operating system (MS-DOS), the general version of its PC-DOS designed for the IBM-PC, became extremely popular for the manufacturers of IBM-PC clones. Of course, DRI attempted again to supplant Microsoft with its new Concurrent PC-DOS, which would run MS-DOS-based software or several CP/M-86-based programs at once. Again, compatibility was finally emerging, even among competitors.

Educational Software: Evolution toward Compatibility

Not only has the educational field become a lucrative target for hardware manufacturers, it has been swamped by demands to do something with computers -- demands to which relatively few educators have seemed able to respond. In 1980, the computer industry seemed to be standardizing with a common disk operating system (the CP/M system) for microcomputers, but then four companies -- Commodore Industries, Tandy Radio Shack, Atari, and Apple Corporation -- rushed to sell education hardware that was in no way compatible. Each had a different hardware and operating system, and they all used different and incompatible forms of the BASIC language. These four companies offered discounts to get their microcomputers into schools, but once the computers were there, the schools were locked into developing software for what they had purchased. Of these four leaders, only Radio Shack offered a form of BASIC that was near to the industry standard version of Microsoft BASIC, MBASIC-80. Apple, with its color graphics, came to dominate most of the educational software pool by the sheer number of programs, good and bad, that had become available.

Fortunately, the IBM-PC, when introduced, did use an extended graphics version of the de facto industry standard, MBASIC-86, that is similar to MBASIC-80. Many developers of vocational education software now write for the IBM-PC first, before attempting to rewrite or convert the programs to run on the APPLE IIe or other systems. The Radio Shack TRS-80 models III and IV will use programs written in IBM BASIC with few, if any, changes if a program does not need to address hardware-specific items. Since many Apple IIe computer users also have installed a Z-80-based accessory card, they too can take advantage of these programs. (At the University of Tennessee, one vocational education software developer, Dr. Walter Cameron, has been very successful in writing complex programs on the Apple III using MBASIC, then transferring the files via a transfer program to the Apple II, TRS-80, IBM-PC, and other computers. In most cases, few if any changes have been required to perform well on all.)

EVALUATING EDUCATIONAL SOFTWARE

The Need for Outside Evaluations

Most teachers can judge instructional content and its usefulness in their classrooms. However, judging microcomputer-based educational software in areas other than instructional content is often infeasible for them, since most instructors do not have the requisite hardware/software experience. Thus, it will probably never be within the competence of most educators to make judgments about the appropriateness, effectiveness, efficiency, operating characteristics, and ease of using and altering educational software.

Instead, to evaluate software, most users will have to rely on others who are skilled in understanding software content and programming standards. Indeed, while it would be ideal to set forth some standard by which software might be evaluated by any user, it is probable that most users would rather rely on an experienced evaluator's opinion. (Even those with expertise often seek external evaluations: for example, the authors of this paper together have many years of experience in operating and programming a variety of microcomputers, but they nonetheless hold off on purchasing an expensive software package until they get an outside opinion on how it compares with other similar packages . . . for once the package is obtained, it may be too late to find out it is not effective or is of little use.)

Types of Software That Need Evaluation

Most of the vocational education software in use has fallen into the category of direct instruction assisted by the computer. Early referred to generically as computer-assisted instruction (CAI), it was even said to yield computer-assisted learning (CAL). As discussed later in the paper, CAI may take the form of drill and practice, tutorials, simulations, games, and a range of short, specialized, tool-like programs that may perform problem-solving tasks ranging from calculating interest to graphically illustrating an electrical series-circuit experiment.

In recent years, general and vocational education software has been developed into a hybrid form that uses data files to record and manage information as well as to assist instruction. Computer-assisted testing (CAT), computer-managed instruction (CMI), registration, planning, student recordkeeping, and a variety of instructional systems using data storage to simulate intelligence all fall into this category.

In recent years, integrated commercial software packages have appeared in many schools and offices. These contain programs for word processing, electronic spreadsheets, text proofreading, and graphic presentations, and they may contain computer-based training programs of the tutorial CAI variety. They teach how to use the computer effectively and thus are important tools in education, especially vocational and technological education. As more and more computer applications find their way into vocational education classrooms, special classes in computer programming and software design will emphasize packages such as program generators, screen generators, graphics systems, computer-assisted design and computer-assisted manufacturing (CAD/CAM), and other machine-language uses.

Aids to Developing and Evaluating Software

The development of programming skill by many teachers will cause a varity of specialized programs to be created. These will be useful and perhaps modifiable by others who understand the programming language used. Generally, however, the programs will have little documentation, so few people other than programmers will know what the programs are supposed to do or how to run them. This problem, be it blessing or curse, means that good documentation writers will still be needed to make the software useful.

Documentation alone, however, cannot make a poorly written program run better or provide better instruction. The technical problems in simply writing the computer programs to make them free of errors in coding, function with proper error traps, and be user-friendly can be major.

An example of one team's assistance to the software profession is a paper presented by Post and Sarapin (1983) at the 1983 Annual Conference of the American Vocational Association. Its title, "Writing and Evaluating Educational Software: Some Key Elements," indicates its value to programmers. While written for the programmer of an Apple II microcomputer, the concepts are sound for any programmer coding in BASIC. Post and Sarapin present a substantial number of short sample programs to illustrate how best to write good code to accomplish certain needs found in most educational software.

Even assuming the programmer is capable of writing good code (not a safe assumption), the content of the program can be difficult to design. Perhaps one of the best sources for general design criteria was prepared by Stuart Crawford (1981): A Standard's Guide for the Authoring of Instructional Software: Reference Manual Volume III. It was published by Joint Educational Management (JEM) Research and is available through the Educational Resources Information Center (ERIC). It covers just about everything from flowcharting through instructional theory to language selection, and it even covers evaluation. Its thoroughness points up the problems inherent in software development and the need for such a document. However, many people writing software may not know of its existence.

One document that is reasonably well known and that has provided a valuable service is The Evaluator's Guide for Microcomputer-Based Instructional Packages. It was prepared at the Northwest Regional Educational Laboratory (NWREL) in Portland, Oregon, by the Microcomputer Software and Information for Teachers (MicroSIFT) group to aid teachers in evaluating microcomputer courseware. It uses an interesting four-phase sifting process which carries the evaluation process from beginning to end. By implication, knowing how and what to evaluate can and should help one understand what is needed to produce good educational software.

Other evaluatory efforts include the following: (1) In 1981, the Educational Product Information Exchange (EPIE) and the Microcomputer Resource Center at Teachers College, Columbia University, began a joint software evaluation project. (2) In 1982, Cohen, in a document entitled

"Evaluating Instructional Software for the Microcomputer" presented to the American Educational Research Association, traces the development of an evaluation instrument; the evaluation procedure; instructional design attributes, including text formats and modes of instruction; and instructional strategies. The forms he includes allow fairly straightforward comparison with the work of other researchers or developers. (3) Another useful document, <u>Evaluation of Educational Software: A Guide to the Guides</u>, was developed by the Southwest Educational Development Laboratory in Austin, Texas in 1983.

The National Center for Research in Vocational Education (NCRVE) has also been involved in or has sponsored efforts in the evaluation of vocational education software. Recently, NCRVE has been involved in a project directed by Shirley Chase to develop an evaluation system for vocational education courseware. The instrument which has been evolved appears similar in ways to that of MicroSIFT. It does not appear to address software other than courseware.

Available Evaluations

There is only so much information that one can glean about a program without using it over a period of time. Certain items do stand out, however, as information one needs before buying educational software. These seem to appear in all educational software evaluations in one form or another. They include the program's title, vendor, price, source address, type of hardware and operating system required, media used, appropriate grade levels, mode of instruction or use, instructional technique used, general objectives, prerequisites, ratings of technical program operation and content presentation, lists of major strengths and weaknesses, and a summary of the evaluation in either narrative or graphic presentation.

Most past evaluations of educational software have been directed toward courseware-type programs. A number of groups, with acronyms for names, have been involved actively in evaluations of this kind of software -- e.g., CONDUIT (University of Iowa), SECTOR (Utah State University), Minnesota Educational Computing Consortium (MECC), and MicroSIFT NWREL. Of these, MicroSIFT may be the most notable: its evaluatory approach has been adopted or adapted by many groups, and it established the aforementioned RICE data base. .

In addition, almost every large software marketing house now does some kind of software review, if not a full-scale evaluation. Some organizations make available, for a subscription fee, catalogs of computer programs written for the educational market. Most of the forms used by them for evaluating educational software seem to be directly related to those developed by MicroSIFT. ("Software sifting" is an increasingly familiar term in the evaluation business. For example, Frankel and Gras [1983] have produced a 254-page guide to software sifting. This guide, while not aimed at the educational market, is such that any serious reviewer who is not familiar with the sifting process should consider acquiring the book.)

Much can be learned about software systems that are of interest by reading the advertising in specialized microcomputer magazines. Many magazines also have large portions of text devoted to software and hardware reviews. For example, one journal, Software Retailing, tells the reader its intent by its title. The January, 1984, issue of PC World provided reviews of over 1,200 software packages that run on the IBM-PC. A vast array of software is listed in Swift's Directory of Educational Software for the IBM-PC. The winter, 1983-84, issue of Classroom Computer Learning provided several pages of descriptive listings of vocational education software as a "Directory of New Educational Computing Software." Similar listings are available for most of the microcomputers commonly used in classrooms. Of necessity, these listings and reviews are short and often pointed, but they give a quick opinion about the software's purpose, effectiveness, and cost.

Need for a Comprehensive Evaluation

Stone (1983, p. 13) suggested five considerations in creating or evaluating educational software. These can be summarized as follows:

- Learning objectives and task analysis
- Use of the technology
- Pedagogical concerns
- Management considerations
- Need for, and content and format of, the accompanying textual material

A single evaluator may not be able to make all the critical judgments necessary. For example, a software designer or programmer -- even a good one -- may not be able to determine the software's appropriate level of instruction or its probable effectiveness in different classroom situations. Some packages may run very well technically but have little useful content. Some flashy software may be fun but be psychologically unsuited to most classroom situations. Learning objectives may be unclear or the textural material not well organized. For these reasons, evaluating a software program may require a group of evaluators.

One problem of large-scale software evaluation efforts can be the lack of composite followup evaluations of software packages. Bias by individual reviewers is almost inevitable. However, the cost of such followup evaluations, particularly in light of the virtual mountain of software now available, can be prohibitive.

Our Approach to Developing an Evaluation Instrument

As suggested above, one of the major reasons for this paper was the need to investigate what had been done in software evaluation and to synthesize the best of the available instruments into a comprehensive instrument. The authors and interested faculty at The University of Tennessee analyzed what was desirable from a user standpoint and then compared that analysis with a variety of evaluation schemes and formats (those of CONDUIT, MicroSIFT, and SECTOR, as well as those of several school districts from around the country as reported in ERIC). From this investigation, we soon determined that a great deal of commonality among evaluatory forms exists. (In fact, as noted above, most of the forms collected seem to have been derived from some common source such as MicroSIFT.) These forms then became the basis for developing the instrument given in Appendix A.

Our evaluatory instrument, and the criteria on which it is based, are discussed later in this paper. But first, it seems appropriate to review the learning theory which underlies (or could underlie) educational software.

LEARNING THEORY TRANSLATED TO EDUCATIONAL SOFTWARE

The use of the computer for education is an outgrowth of behavioral psychology. The behavorial concepts of operant conditioning and the need for immediate feedback and reward underlie most well-designed instructional software. However, through careful presentation techniques, the computer can also fulfill the dreams of the cognitive psychologist by enabling the conceptualization of enormous amounts of material and the application of problem-solving approaches.

Although one cannot prove conclusively how people learn, the relatively young field of neuropsychology, when coupled with the more traditional branches of educational psychology, has begun to arrive at the way people probably learn best. Somewhere between the analytical behaviorist and the synthesizing cognitivist, there is a broad space where behavioral/observable ideas seem to be a subset of more conceptual or perceptual ideas.

Classical and Operant Conditioning

The well-known concepts of stimulus-response-feedback used in classical and operant conditioning appear throughout the literature on the design of computer-assisted instruction. However, as pointed out by Crawford (1981, p. 5), "Since most existing courseware is designed to encourage learning above the reflexive level, classical conditioning appears to have little to do with CAI." The classic Pavlovian reflex training has little relevance to computer-based instruction in any but certain attitude development and behavior modification efforts. Instead, reinforcing complex learning through feedback seems to have a more global effect and fits well with computer capabilities.

The concepts and practices involved in operant, or instrumental, conditioning merit some thought and clarification. The basic notions of rewarding, or reinforcing, behavior through feedback seem to be substantiated in the use of computer-based instruction. Crawford (1981, p. 8) discussed the nature of primary reinforcers (those whose effectiveness is related to their own merit) and secondary reinforcers:

> Secondary reinforcers . . . are only capable of generating a reinforcing effect if the student has previously associated them with other primary reinforcers. For example, spoken praise will only act as a reinforcement if the student has formed an association between it and a physical demonstration of praise (such as a hug). . . .
>
> Reinforcement in CAI need not follow every correct response (continuous reinforcement) but can, instead, be applied discontinuously by means of various reinforcement schedules. In fact, research has demonstrated that while learning takes longer to occur when discontinuous reinforcement is applied, it is less easily extinguished than if learning had occurred as a result of continuous reinforcement.

Hartley and Lovell (1977), reprinted in Walker and Hess (1984, p. 43), cited a number of research studies dealing with computer-assisted learning. One study cited, Anderson, Kulhavy, and Andre (1971, 1972), provides significant insight into how the knowledge-of-correct-results (KCR) type of feedback affects learning:

> The students were given pencil and paper posttests. The results of the experiment showed that all other groups did significantly better than the "no feedback" group. KCR given after wrong responses only was almost as effective as 100 percent KCR, which was the most successful treatment. In a second similar experiment one group was shown the correct response before having

> to type it into the terminal (i.e., a cheat condition). This time the posttests showed . . . the cheat group performing significantly worse than all other groups, even the one which was not given feedback.

Feedback, as used in courseware, is not always a positive reinforcer. Rather, it can merely supply information which enables the user to make corrections or even do nothing. Hartley and Lovell (1977, in Walker Hess, 1984, p. 43) noted in a key paragraph:

> The cause of this might well be the influence of initial work in programmed instruction. Following Skinner (1954), it was supposed that the learning task should be analyzed into steps or tasks small enough to ensure that the probability of a successful response was almost unity. Thus the immediate knowledge of correct results (KCR) would reinforce the learner and strengthen the stimulus-response bond. However, when Grunden (1969) reviewed over thirty-five studies, of which thirteen were concerned specifically with feedback, not one showed a significant response in learning.

While the relationship between feedback and reinforcement may sometimes be difficult to identify as cause and ef'ect, there generally is such a relationship, sometimes positive and sometimes negative. Perhaps, more importantly, CAI needs a variety of feedback modes. Familiarity with the expected reinforcement may make it somewhat ineffective. Intermittent reinforcement will probably produce more effective results than a fixed schedule of reinforcement, be it rate- or interval-based.

Cognitive Approaches

The more Gestalt views of how we learn take a more cognitive and perceptual stance than that of the behaviorists.

With a developing model of how the brain seems to work, a whole new world of instructional strategies has opened up. The theory that the brain's left hemisphere may process information using a sequential logic, leaving simultaneous, or parallel, logic to the right hemisphere, is exciting to ponder. Carl Sagan (1977) popularized this theory in Dragons of Eden. Ornstein (1977) and many others have written extensively in this field. Even the television networks have had programs on some of the more glamorous aspects of this research.

These new theories suggest that learning is a much more complex process than that analyzed and defined by the behaviorist models. The theories also suggest that in traditional teaching based on conventional left-brain-oriented learning theory, we may have overlooked the need to appeal to the intuitive capability of a student's right-brain hemisphere -- the possible source of creative insights. Computer-based instructional strategies could capitalize on these evolving concepts by designing a whole new field of courseware.

Leaders such as Maslow (1954), Bruner (1966), and Piaget (1971) have contributed to the fields of instructional and learning theory, motivation, and the developmental process of cognition. The role of perception in motivation cannot be overstated. For example, a condition perceived as motivating when novel may not be perceived as motivating, and thus may not be motivating, when commonplace. Perhaps the Piagetian idea that intellectual structures are developed by the learner, not taught by the teacher, is sound. If so, the culture surrounding the learner will certainly affect those intellectual structures.

Cultural biases for particular learning styles are also important. For example, in the widespread, traditional learning style which concentrates on rote memory or fact learning, the student develops a fact recall structure but may not develop the relational structures needed for technical problem-solving and systematizing or synthesizing novel solutions. Thus, a CAI program which uses a traditional learning style may cause a student to respond properly by recall but with little practical understanding. The relational structure that allows the student to reorganize, synthesize, and generalize facts into some related whole may go undeveloped.

Papert (1980), in Harper and Stewart (1983, p. 5), after spending years working with and on the LOGO language with children, noted how cultural attitudes toward epistemology interact with learning models:

> I began to see how children who had learned to program computers could use very concrete computer models to think about thinking and to learn about learning and in doing so, enhance their powers as psychologists and as epistemologists. For example, many children are held back in their learning because they have a model of learning in which you have either "got it" or "got it wrong." But when you learn to program a computer you almost never get it right the first time. Learning to be a master programmer is learning to become highly skilled at isolating and correcting "bugs," the parts that keep the program from working. The question to ask about the program is not whether it is right or wrong, but if it is fixable. If this way of looking at intellectual products were generalized to how larger culture thinks about knowledge and its acquisition, we might all be less intimidated by our fears of "being wrong." . . . But thinking about learning by analogy with developing a program is a powerful and accessible way to get started on becoming articulate about one's debugging strategies and more deliberate about improving them.

It may be that, between the behaviorists and the cognitivists, there are great truths hidden in the unknowns. It may also be true that one is looking at the micro and the other at the macro view of how we learn. It is safe to observe, however, that the whole of learning is probably greater than the sum of its parts, and that restructuring derived knowledge can give one new insights not possible by analysis alone.

Learning Efficiency and Effectiveness

Perhaps one of the greatest problems for vocational educators is the problem of retention. The amount of the material retained by the student after a variety of instructional strategies have been tried has been the subject of much research. Conventional wisdom holds that the student will be more apt to retain material if actively involved in the teaching/learning process. A major obstacle to this wisdom is that it does not define the efficiency of the strategy, even though the strategy may be effective with sufficient time. For example, a lecture strategy generally involves presentation of a lesson, applications, and analogies to relate the material to the student's frame of reference. The process may be very efficient but not very effective if the student is not motivated or involved in the learning situation. If the strategy is individualized, as in a CAI system, the discovery process and its applications may be beyond the student's or the lesson's development. For that student — even though he or she is involved in the learning situation — the CAI strategy is neither efficient nor effective without teacher intervention.

Learning efficiency and effectiveness may be related to the cognitive style of the particular student. For example, Charles (1976, p. 50) discusses three styles of learners: the "adventurer," the "ponderer," and the "drifter." The adventurer is likely to be comfortable with latitude in structure and recognition and to need feedback in the form of possible alternatives or perhaps some positive comments. The ponderer seems to need structure and some affiliation and recognition, with feedback in the form of evaluation, corrections, and positive reinforcement. The drifter seems to need structure, affiliation, and recognition to a high degree, with feedback in the form of guidance, urging, and positive reinforcement. This indicates that the adventurer probably would find computer-based learning advantageous. The ponderer would need a more structured form of CAI than the adventurer and more intervention from the instructor. For the drifter, an individual style of learning does not seem appropriate, although small groups could be effective. To achieve high retention levels in all these cases would require very well-designed courseware written by someone who understood the feedback and reinforcement processes.

Malone (1981) presented a theory of intrinsically motivating instruction based on three categories: challenge, fantasy, and curiosity. The challenge must involve the student's self-esteem. The fantasy evokes mental images of things not within the student's experience. Cognitive curiosity is a desire to bring better form to one's knowledge structures. To achieve these, Malone cited the need for variable difficulty levels, multiple level goals, hidden information, and randomness, and for feedback that is surprising and constructive.

Instructional Models

In analyzing instructional models which might be derived from learning theory and combinations of content, time, and expected proficiency levels, Pucel and Knaak (1975, p. 17) suggested eight combinations:

- Fixed content, fixed time, and fixed proficiency
- Fixed content, fixed time, and variable proficiency
- Fixed content, variable time, and fixed proficiency
- Fixed content, variable time, and variable proficiency
- Variable content, fixed time, and fixed proficiency
- Variable content, fixed time, and variable proficiency
- Variable content, variable time, and fixed proficiency
- Variable content, variable time, and variable proficiency

Of these models, the first probably would be inappropriate (if the common learning theories offer any insights). The second is the most often used instructional model for group and even for individual instruction, but for competency-based vocational education, especially, the third model seems to offer the greatest promise. In this type of education, the basic content generally has resulted from a job analysis, and an acceptable level of proficiency has been set. Since students do not learn at a fixed rate, however, it is essential that sufficient time be given to achieve the required proficiency level. The other models all have some potential for courseware, assuming one's objectives are thought out in advance.

In conclusion, it is important to note that a strategy developed for individualized instruction is often equally effective for small-group or even large-group instruction, since groups are actually collections of individuals who learn material differently, and computers allow the responsibility for learning to reside with the learners. In group situations, the number of computer terminals or microcomputers avaliable -- not the instructional strategy -- may be the main constraint.

INSTRUCTIONAL METHODOLOGY FOR COMPUTER DELIVERY

There are probably as many variations on the theme of instructional methodology as there are colleges of education. However, effective methods generally tend to group themselves. So it is with methodologies for computerized instruction. The constraints on using computers for instruction generally come from the nature of the hardware available and the appropriateness of the software to the instructional objectives.

Hardware

Because the computer, or microcomputer, must have input and output devices to interact with the student, hardware considerations need attention. Typically, the output device is a video display, or the output is printed on paper upon command. Recently, the computer's output potential has been extended, at rather low cost, to the plotting of diagrams and to sound, including the spoken word as generated by the computer from internal codes. Subject to the control of the sophisticated programmer, video disks, slide-tape systems, and even video tapes can interact with the computer for a variety of outputs.

Input devices, once restricted to hand-wired connections, switches, key-punched cards, or marked answer sheets, now offer a variety of exciting choices. The typewriter-like keyboard allows direct entry of data to the computer but requires some keyboarding skill. A light pen allows one to simply point to or touch the screen for input. Some microcomputers allow input by simply touching the screen of the video terminal. The "joy stick" and the "game paddle," initially used for video games, are now being applied to graphics and computer-assisted design. A device, dubbed the "mouse," allows one simply to move it on a flat surface while it controls the movement of a spot of light, known as the cursor, on the video screen. Other devices allow one to move a stylus over a "digitizing" tablet to input point coordinates for a drawing or graphic display. For those who can afford the cost, Texas Instruments has introduced voice control of the computer by voice-recognition hardware and software.

Software

Computer-assisted instruction (CAI). Computer-assisted instruction allows the programmer to present and sequence instructional material and, limited only by the hardware and the programmer's ability, to provide technical assistance and feedback. Programs may be short and individual, or they may be sequenced from a menu, permitting exit at any point in the process.

The term CAI is generic and can be confusing because of its many forms. Actually, it might be said that all forms of computerized instruction are CAI. The major CAI modes might include (a) drill and practice, (b) tutorials, (c) games and simulations, (d) computer-managed instruction, and (e) the computer as a tool (e.g., for problem-solving). While some authors such as Steffin (1983) only delineated the first three modes, others such as Stone (1981) pointed out the use of the other modes. Hofmeister (1984) suggested that CAI can be subdivided to include (a) programmed instruction, (b) artificial-intelligence-based CAI, and (c) simulation-oriented CAI.

As the name implies, drill-and-practice programs are supplementary in nature and concentrate on skill-development areas such as math, language, typing, and memory skills. One immediately thinks of flash cards when thinking of drill and practice. At Stanford University, Patrick Suppes, perhaps as much as any other, enhanced the use of drill and practice, especially in the field of mathematics. Entire courses using drill-and-practice CAI are now offered there in mathematics and Russian. Because drill-and-practice programs are relatively simple to produce, novices may produce poorly written versions that confuse the user. For this reason, this type of CAI has been generally criticized by some. Such criticism seems an unwarranted indictment unless it can be shown that most drill-and-practice programs are indeed poorly written.

A tutorial is a CAI program whose purpose is to help the user learn a prescribed set of materials. Tutorials have become extremely useful since they can be tied to specific goals (e.g., how to operate a specific word-processing system). Perhaps some of the best examples of tutorials are those produced commercially to teach users how to work with integrated software packages such as PeachText 5000 or Lotus 1-2-3. Sales of commercial software became dependent on good documentation and tutorials when vendors found that users could not understand the programmers' complex instructions and that these instructions needed to be rewritten by professionals to appear simple and systematic for ease of learning. American Training International (ATI) has produced tutorials that are used for many commercial software packages and that have become models of good tutorial technique.

Simulations, or software imitations of probable conditions, are rather difficult to produce. Rowe (1981), in Walker and Hess (1984, pp. 181-86), described a series of simulations he had reviewed as "pretty poor -- hard to use and hard to understand." He then gave a concise yet thorough set of guidelines for producing simulations. They are summarized as follows:

- Set the simulation parameters well.
- Provide for clear, straightforward questioning and adequate response.
- Use care in preparation and screen display of simulation results.
- Make simulations meaningful, so that they elicit expected behavior.
- Be consistent in overall organization and style.
- Target simulation for particular level of user sophistication.

Hofmeister (1984, pp. 4-11) cited a work by Ellington, Addinall, and Percival (1981, p. 78) which pointed out that simulations in science education could make a valuable contribution in:

> situations where a conventional experiment is either extremely difficult or impossible;
>
> situations where experimental apparatus is either not readily available or too complicated or expensive for general laboratory use;
>
> situations where actual experimental work could be dangerous or would cause unnecessary suffering;
>
> situations where a conventional experiment would take an unacceptably long time to complete.

In the general field of computer-based simulation, games make up a large portion of the software on the market. Of course, in the military, computer-based simulators are used extensively since actual military exercises can be prohibitively costly and dangerous.

Computer-managed instruction (CMI). Since the early 1970s, the development of interactive CMI programs with instructional testing and recordkeeping capabilities have begun to appear for a variety of applications. They were the natural outgrowth of CAI development but could be operated with either a single computer or a roomful of computers. They were intended to free up the teacher to give attention to students.

Generally speaking, CMI is used to manage the recordkeeping of complex instructional systems such as those using competency-based education. It can be used in conjunction with conventional individualized instruction or with appropriate CAI. It can be made to handle delivery of modules or CAI; pretesting, posttesting, and prescription of instruction; and all recordkeeping or gradekeeping. Detailed reports can be generated at any time for the instructor or administrator by using the data base kept by the CMI program.

While recent books have given much treatment to CMI, its programming is quite complex. It can operate well with most interactive-type timeshare minicomputers. Because of the sheer size of the programs and data files, microcomputers without hard-disk drives may have some difficulty if speed is required and the program is not efficiently written. However, most of today's 16-bit microcomputers with 256 kilobytes of random access memory (RAM) can be configured so that the files can be held in main RAM, a feature which can make CMI very efficient and financially within the reach of many classrooms.

In 1975-76, Frantz, Matthews, and Boas (1979; Matthews, 1978), working with Project DELTA at the University of Delaware, developed a three-part

CMI program to handle the delivery of individualized instruction in multioccupational programs for vocational education. It used the BASIC-Plus language and operated on a DEC PDP-11/50 timeshare computer. This system was reviewed by Wang (1982) and was used by him, as part of his dissertation research, as a basis for developing a microcomputer-based CMI system for a TRS-80 Model III. He demonstrated that a full system could work on a microcomputer with two floppy-disk drives, and that it could be used by students with a minimum of training.

Tennyson and Buttrey (1980) presented a comprehensive description of problems associated with CAI without some management control system:

> Instructional research (DiVesta, 1975) and applied projects (Steinberg, 1977) dealing with variables of learner control (using rather large or complex learning tasks) have failed to demonstrate that students can make and carry out decisions related to content elements and personal assessment. Therefore, it appears that program-controlled management systems are necessary for effective computer-assisted instruction.

Hofmeister (1984) devoted an entire chapter to an interesting overview of CMI. In his introduction he made the following point (p. 3):

> While the fortunes of CAI have fluctuated, CMI has been making quiet but substantial contributions to education. With its emphasis on the management of instruction-related information rather than the direct teaching of pupils, CMI may be the most cost-effective example of the applications of computers to instruction.

Producing good educational software. It is absolutely essential that an instruc' nal software package be based on acceptable learning theory or its very inte.' may be scuttled. Good programming may produce wonderful graphics, great sound, and dazzling color but not produce the message or induce learning. In short, each package must have definite learning objectives, use a strategic mix of technology and delivery, and adhere to pedagogy that assures the effectiveness of the package.

A pilot software package should be developed with careful thought, then used with the expected audience to determine needed modifications before being completed and final documentation prepared. Only when the package known to have a sound basis for meeting the expected objectives should it be marketed. However, from the number of poorly written textbooks on the market, it is apparent that marketability is no sure indicator of an educational product's quality -- a caveat that applies to prospective software buyers as well.

Barnes (1984, p. 23) discussed reasons why the marketplace has a lot of software for computer-based education and training that is less than adequate:

> An author familiar with instructional design issues but, ignorant of computer capabilities, will typically produce a "page turner" - a sleep inducing, eye straining manual. In contrast, a technically intelligent individual with scant knowledge of training design will produce a "gee whiz" course that is pure frustration to students.

In further discussing computer-based training, Barnes pointed out several key precepts, summarized below, which good courseware should follow:

- Use effective teaching methods
- Have a sensible flow of material
- Use appropriate testing
- Be engaging, not passive
- Teach what it claims to teach

- Have coverage that meets training-level needs
- Be easy to use
- Use effective screen design
- Use compatible hardware

Barnes also notes that the software should be available from a vendor with a good reputation for service.

This last point raises an interesting issue, however: not much attention has been devoted in educational literature to perhaps the largest development of software, that of programs created by the instructor. The popular computer magazines (e.g., BYTE, 80-Microcomputing, and Creative Computing) have informational articles in this realm. The programmers are dependent on the sketchy documents presented in the articles or on their own ingenuity. The beauty is in the instructor's being master of the software.

However, courseware or management software must be designed with a great deal more sophistication than the average lesson prepared by a classroom teacher. A single courseware package to teach a unit of instruction may take several weeks or months to plan, code, pilot test, and debug before it is ready for effective classroom use. And Alfred Bork (1978) -- noted for his important contributions to the field of CAI, especially in physics -- has commented that it is archaic to think that computer-based materials can be produced by anyone alone (p. 20).

Thus, producing good educational software in most cases will have to be cooperative efforts. There will continue to be a need for courses in technical programming for education, and it will take several hands-on courses. In addition, there is a growing need for coursework which applies good learning theory and instructional theory to the preparation of software. Since not many professors in colleges of education are expert in both the theoretical and the technical areas, it may be some time before the process becomes an ordered array (to use a computer term).

CRITERIA FOR EVALUATING EDUCATIONAL SOFTWARE

To be considered as having merit, any educational software should be well-documented, self-prompting, and easy to use with few unexpected problems, and it should do what it purpor : to do at its prescribed grade level. It should perform well technic. ., using accepted programming techniques. Common shortcomings of educational software involve inappropriateness to the curriculum; content ineffectiveness; inappropriate amount, complexity, or reading level of material covered in each module; lack of user friendliness; high cost; inability to run on most microcomputers; wrong audience addressed; and poor screen presentation.

Generally, the evaluation of educational software is based on a judgment call dependent on the software's characteristics, with the evaluation ranging on a continuum from some level of merit (if a characteristic is present and well-executed) to some level of shortcoming (if the characteristic is absent or poorly executed). With commercially marketed software, the requisite characteristics are almost always present; instead, their quality is the issue for evaluation. For example, the lack of documentation concerning what the program is supposed to do and how it does it would be a definite shortcoming (although usually, as stated previously, documentation is present but may range from excellent to almost useless).

In the various fields of vocational and technical education, it is the more sophisticated business and industrial software and training programs that will complicate the evaluation system. Programs used in computer-assisted design (CAD) and computer-assisted manufacturing (CAM), although they are educational, do not fall into the same categories as CAI and CMI applications. In fact, the former would more likely need evaluation instruments similar to those used for word processing or for electronic spreadsheets, and such evaluations would more likely involve whether a package meets certain minimum standards and how it compares with other such packages.

Nevertheless, it is important to be aware of certain broadly applicable criteria. Those that have been used in most of the available evaluation instruments are presented below as a basis for understanding the evaluation instrument suggested in this paper.

Software Objectives and Design

Perhaps the most important question any software user must ask, before considering any purchase, is, "What is it supposed to do?" This information generally is included in materials accompanying any good software package. Unless the program's purpose is well outlined there, one may not have any idea of the program's value.

Program purpose. In the past, programmers frequently have assumed that a program's purpose would be obvious, so they did not bother to indicate what the program was supposed to do or how to make it do just that. But users need to know more than whether a package is, e.g., a CMI or a tutorial. Instead, they need to know the intended audience and how (well) the package will work for that audience. For example, a typing skills package may do an excellent job for a group of middle school students who need cursory instruction in keyboarding, but it may be totally inappropriate for learning the kind of typing skills needed prior to using a complex word processing package. In addition, the package should contain specific instructional objectives that relate to the proper grade level, and it should indicate whether it is to be used alone or with regular classroom instruction.

Documentation. Proper documentation probably is the weakest aspect of most software. The evaluator, before even attempting to run a program, should read a substantial portion of its documentation and operating instructions. Often these are written by the person who devised the program and understood its operation very well. For this reason, the instructions, although clear to the author, may be obscure to the user.

Good documentation will list the program's objectives, state the prerequisite skills expected, and provide a sample run of the program and/or its expected dialog. Sample screen pictures or diagrams have become customary as a means of leading users through a sample run. If the package is a multipurpose or comprehensive set of programs, then it becomes even more important that documentation be thorough and easy to understand.

Self-documentation. Many programs are self-contained; that is, they are self-documented and obvious as the program runs and thus require less accessory documentation. Some may still be needed, however. For example, with a CMI package, accessory documentation might be needed to answer such questions as: Does the program have textbook correlation? Does it require student worksheets? Are pretests or posttests included in supplemental materials or in the program? Does the program have a "helps" section that can be called when needed?

Hardware/software design. Finally, is the program designed to operate on a variety of microcomputers such as might be available in the classroom? Is the program's hardware requirement commensurate with its cost and usefulness?

Acquisition

One must know where and how to obtain the program. Is a complete description of it available from a supply source? For some time, periodicals have had advertisements for educational computing program packages. However, many good programs aren't advertised in periodicals, since many writers of educational software do not have marketing experience. For this reason, clearinghouses have developed.

Demonstration disks may be required to assess complex programs. If demonstration disks are not available, or if the evaluator has to purchase the software at full cost in order to evaluate it, the program may not get a fair, unbiased evaluation. Generally, most software writers will be happy to have a legitimate reviewer evaluate the program, especially if this may increase sales. For very expensive programs, it is not uncommon to find that a demonstration program that does not allow full use of the program can be purchased for about $10.

Weaver and Holznagel (1984) attempted to identify future trends in available courseware. They pointed out that their RICE database contains information on over 2,400 courseware packages developed by some 300 producers. It was noted that the trend is away from the traditional CAI and toward the computer's use as a tool. Over 40 percent of the vocational education entries, however, were tutorials, with simulations accounting for 20-30 percent. Most of the RICE entries appear to be commercially available packages, not public domain programs.

Cost versus value. Value commensurate with cost is an important criterion that may be overlooked in an evaluation. Sometimes competing packages are available which cost about the same and have similar content but use different instructional approaches and have different advantages and disadvantages. An example of this would be two interactive instructional systems to teach computer-based accounting in high school. One system might be instructionally superior but not allow disk backup without excessive charges. The other might allow easy access to the program so that the teacher could make program changes and as many copies as necessary (within copyright limitations). A program's value versus its cost may be easy or difficult to determine, depending on the circumstances and the evaluator's experience.

Software support. Complex programs inevitably have mistakes, generally referred to as "bugs." These are difficult to find until the program has been run in every conceivable manner; hence, most complex programs go through at least two phases of testing and evaluation before marketing. For this reason, the willingness of the software author or marketer to support the software with updates or corrections at little or no cost is a mark of acceptability. In some cases, especially if additional uses for the program are found in testing, followup suggestions for implementation may become available. This kind of validation of the program's accuracy and support for its use is a very important evaluative criterion.

Program Content

Appropriateness and accuracy. A software program may dazzle with its graphics and screen displays but be ineffective and inappropriate because it is distracting. And, just as some books are highly readable but contain factual errors, so some software may be impressive but misleading because of inaccuracies. These considerations may be difficult to evaluate but should not be overlooked.

Structure for ease of use. A well-designed instructional program will seem to flow naturally, leading the user from segment to segment. It is said to be self-documenting for operation. Its structure should be checked for modularity and logical flow. If the user must key in certain responses, the responses expected should be obvious. If they are not obvious, then the documentation should have pointed them out clearly.

Applications and examples. Examples in the documentation of what to expect in the program must be clear and concise and should have real, not obscure, applications. Good examples further the learning process in a regular classroom and should be expected in a good computer program. (E,g,, one otherwise popular electronic spreadsheet program uses examples of apples and oranges instead of real financial applications. It only serves to confuse many potential users.)

Obsolescence. The nature of the material being presented should be evaluated for currency and probability of becoming out of date quickly. Does it have provisions to allow the user to insert new material? That is, is a program listing provided to make program modification possible?

Pacing rates. The program's pacing must be appropriate for the material and the intended audience. In some instances, it may be appropriate to allow only a prescribed amount of time for a response, but in most cases, user-defined pacing may be preferable. It is possible to accommodate both response types in the same program, but this capability is not often built in.

Response feedback. Regardless of the type of instruction being used, immediate feedback is an important quality since people generally respond well to simple feelings of reward by feedback. A good program will enhance this reward system in subtle ways. Sounds and flashing lights may not be the best alternatives.

Hardware/Software Operation and the User

Error-trapping. There is nothing quite so disturbing as to have a program suddenly cease operation (or, in the language of the trade, to "bomb"). No marketed program should be so poorly tested as to allow accidental program exiting at any point in the program's execution. There must be error-trapping designed to prevent every conceivable error from being accidentally overlooked. This may be one of the most difficult parts of writing a program, but it is essential.

Consider, for example, the penchant of microcomputers to run out of memory available for storing unused character strings (variable values). The strings are no longer used; hence are considered as garbage. If voided in time, the computer eventually may cease to function as it clears its memory by collecting and disposing of all the garbage. It will stop suddenly and just as suddenly start up again -- but unfortunately, this may take several minutes. Meanwhile, panic may have set in, and the user may have aborted the program.

Or consider another example: data put into a program always has limits on the number of acceptable characters. A well-written program will never allow the user to input too few or too many characters. A program that shows the expected number of spaces to be filled should get high marks for efficiency and user friendliness. Other user errors that may occur include attempting to add to a file that does not exist yet and trying to open one that is already open. These errors are easily anticipated and handled by a competent programmer, but other input errors are mistakes that the author either might never think of or might find hard to trap.

Perhaps one error-trapping routine the evaluator should look for is the acceptance by the program of either uppercase or lowercase character responses. Frequently, a program will hang up or fail to respond to an input character if the shift-lock key is not depressed. Programming for this error is not as simple as for some others but can be handled in a subroutine that recognizes either case of character responses by ASCII code value.

User friendliness. Some users simply run a program to see if they can figure out how it works without resorting to reading the "directions." If it runs well, it's user-friendly; if it doesn't, it isn't . . . or so they think. But unless one understands what the program is to do and what is expected, simply running it leaves a massive burden on the author. For example, few people -- even experts -- should expect to sit down and run a complex CMI program without significant instruction concerning its intent and how it is expected to operate. On the other hand, many well-designed tutorial programs can be run with little or no outside assistance.

To ensure user friendliness, many software marketers have developed sample demonstration programs that lead the potential user through the system with sample data in order to demonstrate what can be expected from the package. After such a demonstration, a complex program may be considered to be quite user-friendly. On a simple run of the same program without the demonstration, the program may seem hopelessly complex.

As an evaluatory criterion, then, user friendliness must not be overlooked. In essence, this asks if the program operates smoothly and prompts for expected input without one having to resort to outside help, but often it is the supplementary instructions, not the program itself, that must be user-friendly. Using this criterion thus means one must look at the entire package.

Smoothness of operation. The evaluator needs to determine if the program seems to flow well and logically, with little waiting between steps, especially if nothing is on the screen explaining the wait. For example, if a program is doing some computations that consume more than a few seconds, something appropriate needs to show on the screen. While it is not efficient to interfere with the calculations by having them printed on the screen, having the problem number printed as the results are being calculated let the user feel something exciting is happening. It also gives a feel for how long it takes to accomplish each step. This is especially true of lengthy sorting or searching of files.

Compatibility and Special System Needs

Diskette formats. One perplexing consideration is the availability of the program in a disk format that will operate on the user's computer. For example, if one had a program written for a Radio Shack TRS-80 Model I computer with a single-density disk operating system, it would not run on Apple II microcomputers because of many disk format differences. And even when the computer and program disk are seemingly compatible, there still may be difficulties -- for example, a program may be written to run on the Apple II single-density disk with an early version of Apple's disk operating system, but if the user has a later version, there may be some inconsistencies that could cause the program to malfunction.

And there are further complications: for instance, vendors other than Radio Shack have prepared a multitude of hardware and software enhancements for the TRS-80 series that may make programs not operable with the series' standard disk operating system, TRSDOS.

Programming languages and dialects. Transportability of educational and business software presupposes that interpreters or compilers are available for the language being used by each microcomputer. The BASIC language has a de facto industry standard known as MBASIC-80 and MBASIC-86 developed by Microsoft, Inc. of Bellevue, Washington. This, in one of its versions, generally is supplied with or available on most CP/M-type or MS-DOS/PC-DOS-type microcomputers. With few differences, BASICA, GW-Basic, MS-Basic, and MBASIC-86 are the same language but with a few extensions that are peculiar to certain 16-bit microcomputers. In fact, the TRS-80 systems, using the Z-80 microprocessor, have a dialect of Microsoft BASIC, derived from its version 4.51, which is similar to MBASIC. However, Apple BASIC, with thousands of educational programs; Atari BASIC; Commodore BASIC; and Hewlett-Packard BASIC -- as well as other less well-known BASICs -- have only a cursory resemblance to the de facto standard MBASIC.

The problem of incompatible dialects is not limited to the BASIC language. It is compounded by the variety of dialects for other languages as well -- PASCAL, in particular. Generally, most educational programs are written in BASIC because most common microcomputers have a BASIC interpreter with the system. (Most BASIC programs will not be in compiled form since one would need the compiler run-time system to run the program when purchased. A compiled BASIC run-time system is expensive, and, as an extra-cost item, is not in widespread use.)

Programs written in PASCAL must have the dialect of PASCAL specified. Most dialects are compiled forms and will run only on the system under which they were written. There are at least four major incompatible PASCAL systems. Most educational software users do not have compiler PASCAL run-time systems, just as they do not have compiler BASIC run-time systems.

Programs written in a microprocessor's native assembly language generally are complex programs that provide high-speed operations such as moving screen graphics. They are more complicated to write and may cost considerably more than those written in BASIC, PASCAL, PILOT, or LOGO. The advantages are considerable, however, because they require only the operating system of the computer in order to run. Many business-type programs used in vocational and technical education, such as word processing systems and electronic spreadsheets, are written in assembly language. Such programs should be rated somewhat higher for efficiency and no requirement of extra operating software to run. Again, however, an assembly-language program written for the 8088 microprocessor will not operate on a 6502, 68000, 9900, or, in some cases, the Z-80.

The evaluator does not need to judge the superiority of one language or dialect over another. Instead, what is important is the program's availability in a commonly used variety of disk formats and language dialects to match the equipment found in educational settings. Marketability will help to ensure this: it means that the program must be written and produced for the appropriate variety of microcomputers, or the product may not be widely accepted.

Operating or run-time software. As suggested above, some types of educational software will require special run-time systems. For example, the PILOT language is useful for authoring certain types of CAI programs. In order to run the program, however, one must have the PILOT resident in memory to interpret the program symbols and operate the program.

Programs written in Microsoft MBASIC-80 or MBASIC-86, if properly written, also can be compiled into machine-level code. Once compiled, the program may run five to ten times faster than in the interpreted mode. Generally, however, this type of program is best marketed in the interpreted form with instructions for compiling by those who also have purchased the MBASIC compiler. The compiler may cost as much as $350 for a user license. If the program is compilable, it may have greater value than one not having this capability, but only if speed of operation is important. Many business-oriented programs, such as accounting packages, are written in a compiled BASIC. In all probability, compiled forms of educational software have not developed rapidly because of the cost of the support compiler and little need for speedy file handling. But if the program is one involving data-file handling, speed may be extremely important, especially where sorting and searching is involved.

THE EVALUATION INSTRUMENT

One of the major reasons for this paper was to synthesize the best of the instruments available for evaluating vocational-technical education software. As indicated previously, the authors and interested faculty colleagues reviewed the evaluation approaches of various agencies and school districts. It was then decided to combine the best features of these approaches and to evolve a single form with several sections. (It appears that Chase, Gordon, and Makin [1984] moved through a similar process in their recently completed project at the National Center for Research in Vocational Education [NCRVE].)

The form and substance of the final instrument given in Appendix A is thus derived from the excellent work of the MicroSIFT group and others cited earlier. While new evaluation instruments will no doubt surface, as has the one just developed by the NCRVE, most will, of necessity, address the same questions.

For all practical purposes, this instrument is applicable to almost any educational software as well as to vocational-technical education software. The latter is distinct in that it includes the more business-oriented software for CP/M-type and MS-DOS/PC-DOS-type computers; otherwise, it is similar to educational software in general. For management and certain specialized software such as CAD/CAM, word processing, and electronic spreadsheets, however, a marked difference is apparent, both in software format and in the capability of the hardware used.

Examples of how the instrument can be applied are included as

Appendix B. As illustrated there and as described below, the instrument uses a checklist format with rating scales. Responses are numerically rated to enable statistical analysis and eventual conversion to a computer-based inventory.

Description of the Instrument

Since the instrument was to be general in nature, it was subdivided into five general parts, with completion of the third or fourth dependent on the kind of package being evaluated:

- General descriptors of the software package
- Documentation available
- Evaluation of courseware-type programs
- Evaluation of management-type programs
- Recommendations, strengths, and weaknesses

The first part is subdivided into categories giving (a) the program title, producer and vendor, and program type, (b) the mode or purpose of the package; (c) pricing versus what comes with the package; (d) the updating policy; (e) the types of hardware required for proper operation; and (f) the software support needed (i.e., the type of operating system or language required).

Available documentation is covered by the second part, which has two sections. One deals with how the documentation is provided; the other rates the documentation.

The actual checklist/rating parts, often considered the heart of the evaluation, are provided for two types of software: courseware and management programs. In addition to rating scales, space is provided for general comments and observations.

Recommendations and space for narratives on general strengths and weaknesses and other comments conclude the instrument. This last part, when used in conjunction with the rating scales, should provide the potential user with reasonable judgments concerning the assets, limitations, and overall quality of the package.

Pilot-Testing

The instrument was pilot-tested in the microcomputer laboratory of the Department of Technological and Adult Education at The University of Tennessee, Knoxville. With early versions, evaluators were unable to make some judgments due to unclear or inappropriately sequenced statements in the ratings. Modification of the instrument removed most of the problems.

From the pilot-testing, it became obvious that people not familiar with software design would have trouble evaluating a complex software package. The major problem was their lack of understanding of how the author programmed the package or what was intended by the program's operation. Some -- but not all -- of this difficulty stemmed from inadequate documentation or poorly designed software. It thus appears that regardless of the evaluation instrument, software evaluation is becoming a field for advanced computer users with an understanding of programming structures.

CONCLUSION AND RECOMMENDATIONS

As the microcomputer becomes increasingly present in the educational marketplace, software problems will be compounded. Two events may ease these problems somewhat, although not entirely. First, the attempts to arrive at a set of standards for the BASIC language by its original developers may become reality. If so, the language will have routines that tend to be less machine-specific. Second, the trend toward compatibility in hardware, as evidenced by the IBM clones that are flooding the market, may continue (although this could also have the negative effect of stifling advances in hardware and software).

As more colleges of education, public schools, independent schools, homes, and military operations develop their appetites for computers, the demand for more complex software will explode. More vendors will surface in the marketplace, with new software capabilities. For example, techniques will emerge for writing programs by using program generators, creating a new instructional need for those who wish to use the program generators. In short, such exciting developments will help to solve existing problems but will also lead to new ones.

Thus, more attempts to set standards for educational software will be needed. Centers for evaluation will have to be funded to review what is appearing on the market. A primary thrust of these efforts would be to keep the unsuspecting school user from investing money in low-quality products. Teachers, along with other users, will tend to rely on what are considered to be experts to determine what constitutes good educational software. Us [illegible] will decide which software packages are appropriate for them, but the actual evaluation of complex software will probably be left to experts.

In the last analysis, however, it is the teacher who makes the classroom effective, and it is the teacher's involvement in computer technology and software design and use that will make the difference in good educational software.

Recommendations

The following are a few brief recommendations by the authors, based on their assessment of probable trends in computer software:

- The demand by homes for computer software will grow rapidly, perhaps ahead of the schools' demand. Leaders need to be aware of this trend and provide guidance for software development.
- Research is needed on how instructional software can be designed to enable students with various backgrounds to retain and use what they have learned. This is especially true at the upper grade levels and in adult education, where literacy is a major problem.
- Colleges of education need to invest in the development of their faculties as resources to help design good instructional software.
- The infant world of artificial intelligence (AI) is now coming of age with the developing computer AI languages. These will enable new directions in instructional theory and design, especially if educators become involved in this development.
- Vocational and technical education must take the lead in directing the general education of all children toward applications-oriented secondary and post-secondary education. The computer offers a tool for this if the necessary software can be developed.

APPENDIX A:

EDUCATIONAL MICROCOMPUTER SOFTWARE DESCRIPTION AND EVALUATION FORM

EDUCATIONAL MICROCOMPUTER SOFTWARE DESCRIPTION AND EVALUATION FORM

I. **THE SOFTWARE PACKAGE**

A. **Package:** Title____________________ Version______

Copyright: Yes____ No____ Date__________ Public Dom.____

Producer____________________ Vendor__________

Vendor address______________________________

City______________ State___ Zip______ Phone__________

Vocational area______________ Subject area__________

ERIC/RICE descriptors______________________________

B. **Package Type:** Single prog.___ Prog. cluster___ Integrated cluster___

C. **Purpose:** Administrative____ Instructional mgmt.____ Data mgmt.____

CAI____ Writing____ Special purpose______________

What is program supposed to do?______________________

____

D. **Mode:** Assessment__________ Enrichment_____ Simulation____ Game____

Problem Solving______ Drill and practice_______ Tutorial_______

Computer as a tool___ Record keeping____ Other______________

E. **Pricing/Cost:** Initial copy______ Multiple copies_____ Updates______

Backup approved: Yes____ No_____ Program locked: Yes____ No____

Field test available: No_____ On request____ With package____

Installation assistance: Yes____ Cost_______ No____

Staff training: Yes____ Cost_______ No____

F. **Update policy:**____________________ Toll-free phone__________

G. **Hardware required (or will work with):** _____KBytes memory

Computer: IBM-PC_____ Apple II_____ TRS-80 III/4_____ TI-99/4_____

Atari______ COM PET______ COM 64_____ Other______________

Storage medium: Cass. tape_____ 5.25" disk____ 8" disk____ ROM_____

Number Req.: SS/SD disk dr.____ SS/DD disk dr.____ DS/DD disk dr.____

Monitor: 40 Col.____ 80 Col.____ Color TV____ Color Comp____ RGB____

Other: Printer____ Joy Stick/Paddle____ Light pen____ Modem____

Digitizer____ Koala pad____ Mouse____ Plotter____ Opscan____

H. **Software support:** Language or run-time system____________ Version____

Operating systems: PC-DOS____ MS-DOS____ APPLEDOS____ TRSDOS____
(Give Version #)
CP/M____ Other____ ____________________

II. **DOCUMENTATION AVAILABLE:** Please check appropriate blanks under, **P** if in the program, **S** if in supplementary materials, **NI** if not included, **NA** if not applicable.

P	S	NI	NA		P	S	NI	NA	
				Grade/ability level					Index
				Instructional objectives					Table of Contents
				Prereq. skills/Activities					Purpose of package
				Samp. prog. output/dialog					System overview
				Operating instructions					Intended audience
				Pretest					System capacity
				Posttest					Hardware information
				Teacher's information					Operating instruction
				Resource/ref. information					Help Section
				Student's instructions					Sample screen display
				Student worksheets					Sample report
				Textbook correlation					Sample graphs
				Followup activities					Sample plots
				Program listing					Interface instruction

Please check blanks for appropriate rating, i.e., **0** (Not applicable), **1** (Disagree), **2** (Neutral), **3** (Agree), **4** (Strongly Agree).

0	1	2	3	4		**COMMENTS**
					1. Documentation is easy to read, clear and straightforward.	
					2. Documentation is well organized.	
					3. Examples are relevant to expected use.	
					4. Printed documentation provides assistance for easy program use.	
					5. Documentation, displays, and reports consistent in format and terminology.	

III. **COURSEWARE-TYPE PROGRAMS:** (CAI, simulations, problem solving, or tool)

A. Intended audience:______________________________

B. Instructional grouping: Individual___ Group size: Small___ Large___

C. Objectives: ___stated ___inferred ___not included

D. Prerequisites: ___stated ___inferred ___not included

E. Describe package content and structure, including any record keeping or reporting functions.

F. Please check blanks for appropriate rating, i.e. 0 (Not applicable), 1 (Disagree), 2 (Neutral), 3 (Agree), 4 (Strongly agree).

Content Characteristics **COMMENTS**

0	1	2	3	4		
___	___	___	___	___	1.	Content is accurate.
___	___	___	___	___	2.	Content has educational value.
___	___	___	___	___	3.	Content is free of racial, sex, ethnic, or other stereotypes.

Instructional Characteristics **COMMENTS**

0	1	2	3	4		
___	___	___	___	___	1.	Purpose of program or package is well defined.
___	___	___	___	___	2.	Defined purpose of package is achieved.
___	___	___	___	___	3.	Objectives of program are clear to the user.
___	___	___	___	___	4.	Content is presented clearly and logically.
___	___	___	___	___	5.	User responses are clearly prompted.

___	___	___	___	___	6.	Calculations or outputs are accurate and easy to understand.
___	___	___	___	___	7.	Difficulty level is appropriate for target audience.
___	___	___	___	___	8.	Sound teaching/learning strategies are used in the program.
___	___	___	___	___	9.	Content and operation of the package is motivational.
___	___	___	___	___	10.	Creativity is stimulated by use of this program or package.
___	___	___	___	___	11.	Feedback on student response is employed effectively.
___	___	___	___	___	12.	Learner can control rate and sequence of content and review.
___	___	___	___	___	13.	Instruction is integrated with previous student experience.
___	___	___	___	___	14.	Learning can be generalized to an appropriate range of situations.
___	___	___	___	___	15.	Effectively uses outside materials.
___	___	___	___	___	16.	Student interaction is sufficiently varied to maintain interest.
___	___	___	___	___	17.	Sequencing of topics makes sense or follows acceptable strategy.
___	___	___	___	___	18.	Adequate review of material is provided.
___	___	___	___	___	19.	Graphics, color, or sound are appropriate for instruction.

IV. TECHNICAL CHARACTERISTICS (All Programs)

COMMENTS

0	1	2	3	4		
___	___	___	___	___	1.	Intended users can operate the program easily and independently.
___	___	___	___	___	2.	Teachers can utilize the package with little difficulty.
___	___	___	___	___	3.	Program uses computer capabilities appropriately and effectively.
___	___	___	___	___	4.	Program is reliable in normal use.
___	___	___	___	___	5.	Program has adequate error trapping.
___	___	___	___	___	6.	Information displays are effective.
___	___	___	___	___	7.	Screen displays are easy to read and free of confusion.
___	___	___	___	___	8.	Program output allows appropriate generation of results to hard copy.
___	___	___	___	___	9.	Timing loops or pauses are adequately controlled.
___	___	___	___	___	10.	Responses may be counted and/or displayed for effective feedback.
___	___	___	___	___	11.	Program can be modified with little or expected effort.

___	___	___	___	___	12. User support materials are effective.
___	___	___	___	___	13. User support materials are sufficiently comprehensive.

V. ADMINISTRATIVE, INSTRUCTIONAL (CMI), OR DATA-MANAGEMENT PROGRAMS:

A. Intended audience ____

B. Please check blanks for appropriate rating: e.g., 0 (Not applicable), 1 (Disagree), 2 (Neutral), 3 (Agree) or, 4 (Strongly agree).

Inputs and Operations

0	1	2	3	4		COMMENTS
___	___	___	___	___	1. Package is menu driven.	
___	___	___	___	___	2. Program runs at acceptable speed.	
___	___	___	___	___	3. User informed of program malfunctions and given instructions for restart.	
___	___	___	___	___	4. Package provides for easy or automatic restart or recovery.	
___	___	___	___	___	5. Data entry prompts are adequate.	
___	___	___	___	___	6. Users can define student IDs or code options	
___	___	___	___	___	7. Data entry procedures are consistent from module to module.	
___	___	___	___	___	8. Editing options after each data set entry are adequate.	
___	___	___	___	___	9. The package permits entry only of acceptable data--good error trapping.	
___	___	___	___	___	10. Optional user defined data fields have been provided for adequately.	
___	___	___	___	___	11. Input field sizes are adequately described.	
___	___	___	___	___	12. Data fields are designed to accommodate sizes of data used in schools.	
___	___	___	___	___	13. Capacity of system and individual records suitable for intended use.	
___	___	___	___	___	14. Records are easily retrieved by record number.	
___	___	___	___	___	15. Records are easily retrieved by contents, e.g., name or Soc. Sec. #.	
___	___	___	___	___	16. Can easily determine number of records already used.	
___	___	___	___	___	17. Package provides adequate file maintenance capabilities.	
___	___	___	___	___	18. Adequate warning before file capacity is reached.	
___	___	___	___	___	19. Existing capacity to copy data files from package is sufficient.	

Outputs

COMMENTS

0	1	2	3	4	
___	___	___	___	___	1. Screen displays are easy to read and use.
___	___	___	___	___	2. Program provides outputs appropriate for the program's purpose.
___	___	___	___	___	3. Program provides suitable options for screen display and hard copy.
___	___	___	___	___	4. Reports are easy to use, with good spacing & meaningful abbreviations.

Management Rating

COMMENTS

0	1	2	3	4	
___	___	___	___	___	1. Package is easy to install and use.
___	___	___	___	___	2. Package provides significant advantage over manual methods.
___	___	___	___	___	3. Program is flexible and can be adjusted to loca needs.
___	___	___	___	___	4. Data generated by one program can be accessed by other programs in series.
___	___	___	___	___	5. Program is reliable.
___	___	___	___	___	6. Package does the tasks it claims to do.
___	___	___	___	___	7. Software provides for protection against access to confidential data.
___	___	___	___	___	8. Software provides safeguard against inadvertent loss of data files.

C. **General Comments and Observations:**

1. Purpose:

2. Describe system capacities:

3. List or describe possible outputs:

4. List or describe major components:

5. Other possible applications:

VI. **RATING SUMMARY** Please rate: 0 (Low) to 4 (High) for each item.

0	1	2	3	4	
___	___	___	___	___	1. Content fulfills purpose
___	___	___	___	___	2. Instructional characteristics
___	___	___	___	___	3. Technical characteristics
___	___	___	___	___	4. Management capability as stated

5. Describe the potential use of this package in the classroom.

6. Estimate the amount of time a student would need to work with this package to achieve its objectives. Suggest total time, time/day, or time range.

VII. **RECOMMENDATIONS:** All software types
Please check the appropriate recommendation below.

_____ 1. I recommend this package.

_____ 2. I would recommend this package with few if any changes. (See suggestions below.)

_____ 3. I would recommend this package only if certain changes were made. (Note changes under weaknesses below.)

_____ 4. I would not use or recommend this package. (Note reasons under weaknesses below.)

A. Major Strengths:

B. Major Weaknesses:

C. Other Comments:

APPENDIX B:

SAMPLE EDUCATIONAL SOFTWARE EVALUATIONS

EDUCATIONAL SOFTWARE EVALUATION

TITLE: Business Package I **VER:** 1.0 **COPYRIGHT:** Yes **DATE:** 1980
VENDOR: Micro Learningware
ADDRESS: Highway 66 South, Box 307
CITY: Mankato **ST:** MN **ZIP:** 56002-0307 **PH:** 507-625-2205
PKG TYPE: Program Cluster **VOC AREA:** Bus. Educ. **SUBJ AREA:** Accounting
PURPOSE: CAI
PROGRAM IS SUPPOSED TO: Compute amortization, depreciation, and bank reconciliation. Simulates stock market operation and allows two students to participate. Assists instructor in explaining variable and equation manipulation.
MODE: Simulation
PRICE 1ST COPY: $39.95 **MULT COPIES:** None **UPDATES:** none indicated
BACKUP OK: No **PROGRAM LOCKED:** Yes **FIELD TEST AVAILABLE:** No
INSTALLATION ASSISTANCE: None **STAFF TRAINING:** No
COMPUTER USED: AppleII with 48K memory
HARDWARE REQ'D: 1 SS/SD 5.25" disk drive, 40 column monitor or TV, printer
SOFTWARE SUPPORT REQ'D: AppleDOS Ver. 3.3 with AppleSOFT BASIC
DOCUMENTATION: P=In program, S=In suppl. materials, NI=Not incl., NA=Not appl.

Grade/Ability:	NI	Instr Obj:	NI	Prereq skill/act:	NI	Samp prog output:	NI
Operating Inst:	NI	Pretest:	NI	Posttest:	NI	Teacher inform:	NI
Resource/ref info:	NI	Student instr:	NI	Student wkshts:	NI	Text correlation:	NI
Followup activity:	NI	Program listing:	NI	Index:	NI	Table of content:	NI
Purpose of pkg:	NI	System overview:	NI	Intended aud:	NI	System capacity:	NI
Hardware info:	NI	Help section:	NI	Sample screen:	NI	Sample report:	NI
Sample graphs:	NI	Sample plots:	NI	Interface instr:	NI		

DOCUMENTATION RATING: (0=Not included, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly Agree)
Readability: 0 Organization: 0 Examples relevant: 0 Provided assist: 0
Consistent in format and terminology: 0 (Package comes with a very short set of instructions and no documentation with which to determine anything about the program.)

COURSEWARE EVALUATION:
Intended audience: Not stated, but probably secondary/post secondary level.
Objectives: (Inferred as useful to supplement classroom discussion of financial equations and their manipulation.)
Prerequisites: (Inferred) algebra
Package Description: A cluster of programs divided into five sections which are menu driven. Outputs may be directed to the printer. The printer output section formats results of computations into reports for each program segment.

RATINGS: (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree)
Content: 3.0 **Accuracy:** 3 **Educational Value:** 2 **Freedom from biases:** 4

Instructional Characteristics: 1.5

Purpose defined:	1	Purpose achieved:	1	Objectives clear:	1	Presented clearly:	1
Response prompts:	3	Outputs make sense:	3	Diff level approp:	1	Learning strategy:	1
Content motivate:	1	Creativity stimula:	1	Feedback helpful:	1	Learner control:	3
Relate prev exp:	2	Generalizability:	2	Use outside mater:	0	Interact varied:	1
Sequence topics:	1	Adequate review:	1	Appropriate use of graphics/color:	3		

TECHNICAL CHARACTERISTICS: 2.0

User operate OK:	3	Teacher can use:	3	Comp use appropr:	3	Reliability:	3
Errors trapped:	4	Effective display:	3	Displ easily read:	3	Give good hd copy:	4
Pauses appropr:	4	Responses counted:	2	Modified easily:	1	Supp mater good:	1

User support materials comprehensive: 1

RATING SUMMARY: 2.0
Purpose fulfilled: 1 Instructional characteristics: 2 Tech characteristics: 3
Management characteristics: 0

Potential uses in classroom: It could be used to assist the teacher in demonstrating examples of the manipulation of a financial equation to derive values.

Time needed to complete package: An hour for each module.

RECOMMENDATIONS: Package not recommended for educational use.
Major Strengths: None apparent
Major Weaknesses: Lack of documentation or suggestions for use. Software seems to have been written for an application other than for education.

COMMENT: Package could be made more effective by the inclusion of good documentation and instructional strategies.

EDUCATIONAL SOFTWARE EVALUATION

TITLE: Information Master **VER:** 5.3 **COPYRIGHT:** Yes **DATE:** 1979
VENDOR: High Technology Software, Inc. **AUTHORS:** James Cox & Steven Williams
ADDRESS: P.O. Box 60406
CITY: Oklahoma City **ST:** OK **ZIP:** 73146 **PH:** 405-524-5249
PKG TYPE: Integrated cluster **VOC AREA:** All **SUBJ AREA:** Data Management
PURPOSE: Data management
PROGRAM IS SUPPOSED TO BE: A control program for data management that organizes, schedules, manipulates data, and generates reports.
MODE: Record keeping
PRICE 1ST COPY: $150 **MULT COPIES:** None **UPDATES:** Within 90 days
BACKUP OK: No **PROGRAM LOCKED:** Yes **FIELD TEST AVAILABLE:** No
INSTALLATION ASSISTANCE: Yes by phone **STAFF TRAINING:** No
COMPUTER USED: AppleII with 48K memory
HARDWARE REQ'D: 2 SS/SD 5-1/4" disk drives, printer, 40-col monitor
SOFTWARE SUPPORT REQ'D: APPLEDOS 3.3 with BASIC

DOCUMENTATION: P=In program, S=In suppl. materials, NI=Not incl., NA=Not appl.

Grade/Ability:	S	Instr Obj:	NA	Prereq skill/act:	NA	Samp progr output:	S
Operating Inst:	S	Pretest:	NA	Posttest:	NA	Teacher inform:	S
Resource/ref info:	S	Student instr:	S	Student wkshts:	S	Text correlation:	S
Followup activity:	NA	Program listing:	NI	Index:	S	Table of content:	S
Purpose of pkg:	S	System overview:	S	Intended aud:	S	System capacity:	S
Hardware info:	S	Help section:	S	Sample screen:	S	Sample report:	S
Sample graphs:	NA	Sample plots:	NA	Interface instr:	S		

DOCUMENTATION RATING: 4.0 (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly Agree)

Readability:	4	Organization:	4	Examples relevant:	4	Provided assist:	4
Consistent in format and terminology:	4						

TECHNICAL CHARACTERISTICS:

User operate OK:	4	Teacher can use:	3	Comp use appropr:	4	Reliability:	4
Errors trapped:	4	Effective display:	4	Displ easily read:	3	Give good hd copy:	4
Pauses appropr:	0	Responses counted:	4	Modify easily:	0	Supp meter good:	4
User support materials comprehensive:	4						

ADMINSTRATIVE, CMI, OR DATA MANAGEMENT PROGRAMS:
Intended audience: May be used by instructors, administrators, or students who need actual experience with professional record-keeping programs.

RATINGS: (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree)
Inputs and Operations: 3.95

Pkg menu driven:	4	Runs at good speed:	4	Informs on errors and restart:	3
Easy prog restart:	4	Data prompts OK:	4	Users can define code options:	4
Data entry consist:	4	Editing opts adeq:	4	Permit entry of good data only:	4
Opt user def fields:	4	Input fld size descr:	4	Data fld size OK for school use:	4
Sys & rec capac OK:	4	Rec retrieve by #:	4	Rec easy to retrieve by content:	4
Find # of recs used:	4	Adequate file maint:	4	Warns when capacity is near:	4
Can copy data files:	4				

Outputs: 3.5
Disply easy to read:4 Approp for purpose: 3 Option for screen or hard copy: 4
Reports easy to use:3

Overall Mgmt Rating: 4.0
Easy install & use: 4 Advant over manual meth: 4 Flexible for local needs:4
Data generated by progr accessible by other programs: 4 Program is reliable: 4 Does what claims to do: 4
Confidential data protected: 4 Loss of data protected: 0

General Observations:
Purpose: Data base file manager. Controls input and retrieval of data.
System capacities: 1000 records, 20 fields/record, 99 characters/field, five sorts w/6 keys per sort, 15 report formats, 15 columns per format.

Possible outputs: string or numerical in user defined formats, e.g., class schedules, textbook records, library lending records, etc.

Major components: System configuration, file creation, file editing, report printing, file sorting

Other possible applications: Limited by imagination.

RATIN SUMMARY:
Purpose fulfilled: 4 Instructional characteristics:3.8 Tech characteristics:
Mgmt characteristics:4

RECOMMENDATIONS: Recommended for AppleII users or instuctors needing well documented and easy to use applications in data-base management.
Major Strengths: Ease of use, relatively low cost, excellent documentation.
Major Weaknesses: Highly structured, with some limits on output printing capabilities.

Comments: A commercial program that could have excellent use in an instructional simulations as well as in real life applications.

EDUCATIONAL SOFTWARE EVALUATION

TITLE: Job Control System **VER:** Demo **COPYRIGHT:** Yes **DATE:** 1983
VENDOR: High Technology Software Products, Inc. **Author:** Mark Nettleingham
ADDRESS: P. O. Box 60406
CITY: Oklahoma City **ST:** OK **ZIP:** 73146 **PH:** 405-524-5249
PKG TYPE: Integrated cluster **VOC AREA:** Ind.Ed./Bus. **SUBJ AREA:** Data proc. Shop mgmt.
PURPOSE: Data management
PROGRAM IS SUPPOSED TO: Operate a job-control system for work in progress, profit and loss, cost estimating, etc. for small-sized company.
MODE: Record management
PRICE 1ST COPY: $450 (AppleII) **MULT COPIES:** $450 **UPDATES:** Winthin 90 days
BACKUP OK: One copy **PROGRAM LOCKED:** Yes **FIELD TEST AVAILABLE:** No
INSTALLATION ASSISTANCE: Yes, by phone **STAFF TRAINING:** No
COMPUTER USED: AppleII w/64K memory, AppleIII w/128K, & IBM-PC w/128K
HARDWARE REQ'D: 2-SS/SD 5-1/4" Disk drives, monitor, printer, language card
SOFTWARE SUPPORT REQ'D: APPLEDOS 3.3 w/PASCAL

DOCUMENTATION: P=In program, S=In suppl. materials, NI=Not incl., NA=Not appl.

Grade/Ability:	NA	Instr Obj:	NA	Prereq skill/act:	NI	Samp progr output:	S
Operating Inst:	S	Pretest:	NA	Posttest:	NA	Teacher inform:	S
Resource/ref info:	NI	Student instr:	NA	Student wkshts:	S	Text correlation:	NA
Followup activity:	NA	Program listing:	NI	Index:	S	Table of content:	S
Purpose of pkg:	S	System overview:	S	Intended aud:	S	System capacity:	S
Hardware info:	S	Help section:	S	Sample screen:	S	Sample report:	S
Sample graphs:	NA	Sample plots:	NA	Interface instr:	S		

DOCUMENTATION RATING: 4.0 (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly Agree)
Readability: 4 Organization: 4 Examples relevant: 4 Provided assist: 4
Consistent in format and terminology: 4

TECHNICAL CHARACTERISTICS: 3.5

User operate OK:	3	Teacher can use:	3	Comp use appropr:	3	Reliability:	4
Errors trapped:	4	Effective display:	4	Displ easily read:	4	Give good hd copy:	4
Pauses appropr:	4	Responses counted:	0	Modified easily:	1	Supp mater good:	4

User support materials comprehensive: 4

ADMINSTRATIVE, CMI, OR DATA MANAGEMENT PROGRAMS:
Intended audience: Post-secondary education indutrial or business education

RATINGS: (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree)
Inputs and Operations: 3.8

Pkg menu driven:	4	Runs at good speed:	3	Informs on errors and restart:	4
Easy prog restart:	4	Data prompts OK:	4	Users can define code options:	0
Data entry consist:	4	Editing opts adeq:	3	Permit entry of good data only:	4
Opt user def fields:	4	Input fld size descr:	4	Data fld size OK for school use:	4
Sys & rec capac OK:	4	Rec retrieve by #:	4	Rec easy to retrieve by content:	4
Find # of recs used:	4	Adequate file maint:	4	Warns when capacity is near:	4
Can copy data files:	3				

Outputs: 3.75
Disply easy to read:4 Approp for purpose: 4 Option for screen or hard copy: 3
Reports easy to use:4

Data Management Rating: 3.8

Easy install & use: 4	Advantage over manual meth: 4	Flexible for local needs:4
Data generated by progr accessible by other programs: 0	Program is reliable: 4	Does what claims to do: 4
	Confidential data protected: 0	Loss of data protected: 3

General Observations:
Purpose: To monitor current status of any job in terms of cost and time. Daily and total costs based on custom-designed reporting and configuration. Seven categories compare actual-estimated costs. Completed jobs are audited to determine profitability.

System capacities: AppleII version has 100-job capacity

Possible outputs: Work orders for 50 cost centers, jobs/numerical, jobs/due date, jobs/detail (cost breakdown), job cost summary for 7 major categories.

Major components: Contains PASCAL editor, assembler, and compiler
Other possible applications: Can assist in tracking, computing, audits job progress. Can be applied to service, process, or piece work applications.

RATING SUMMARY:
Purpose fulfilled: 4 Instructional characteristics: 3 Tech characteristics: 4
Management charac: 3.8
Potential uses in classroom: It is a functional job control system that could provide excellent simulation if implemented properly by the instructor.

Time needed to complete package: Depends on the complexity of the simulation the instructor would build into the simulation.

RECOMMENDATIONS: Recommend package as is for instructors who wish to demonstrate or simulate a functional job control system.

Major Strengths: Quality of documentation and applicability
Major Weaknesses: Probably was designed with an actual user in mind with no real thought about its educational applications. Documentation is aimed at the business/industrial user.

Other Comments: Screens are displayed quickly, clearly, and are well organized. It is a very professional demonstration package.

EDUCATIONAL SOFTWARE EVALUATION

TITLE: Microcomputer Applic. in Agric. **VER:** 1.0 **COPYRIGHT:** Yes **DATE:** 1984
VENDOR: Mid-America Voc. Curric. Consortium, Inc. **CONTACT:** Jane Huston
ADDRESS: 1500 West seventh
CITY: Stillwater **ST:** OK **ZIP:** 74074 **PHONE:** 405-377-2000 X401
DESCRIPTORS: agriculture, microcomputer, CAI
PKG TYPE: Integrated cluster of programs **VOC AREA:** Agriculture
PURPOSE: CAI **Subj Area:** Agribus.; Ag. Mech.; Animal Sci.; Crop Sci.; Hort.
PROGRAM IS SUPPOSED TO: Provide instruction on use of microcomputer and agricultural applications in above areas. Provides transparencies for use by teacher and good documentation including objectives and outlines for all modules. Evaluation components including posttests and answers are in package.
MODE: Problem solving, computer as a tool, tutorials
PRICE 1ST COPY: $75 **MULT COPIES:** **UPDATES:**
BACKUP OK: Yes **PROGRAM LOCKED:** No **FIELD TEST INFO AVAILABLE:** No
INSTALLATION ASSISTANCE: None indicated **STAFF TRAINING:** Not needed
COMPUTER USED: AppleII or TRS-80 III/IV
HARDWARE REQ'D: 5-1/4 disk drive and monochrome monitor
SOFTWARE SUPPORT REQ'D: AppleDOS 3.3 & BASIC or TRSDOS 1.3 and BASIC

DOCUMENTATION: P=In program, S=In suppl. materials, NI=Not incl., NA=Not appl.)

Grade/Ability:	NI	Instruct Obj:	S	Prereq skill/act:	NI	Samp prog output:	S
Operating Instr:	P-S	Pretest:	NI	Posttest:	S	Teacher info:	S
Resource/ref info:	S	Student instr:	P-S	Student wkshts:	S	Text correlation:	N
Followup activity:	S	Progr listing:	P-S	Index	NA	Table of content:	S
Purpose of pkg:	S	System overview:	S	Intended aud:	S	System capacity:	NA
Hardware info:	S	Help section:	NA	Sample screen:	NA	Sample report:	NA
Sample graphs:	NA	Sample plots:	NA	Interface instr:	NA		

DOCUMENTATION RATING: 3.4 (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly Agree)
Readability: 3 Organization: 4 Examples relevant: 3 Provided assistance: 4
Consistent in format and terminology: 3

COURSEWARE EVALUATION:
Intended audience: Sec./Post Sec. Agriculture **Size:** Individual/small group
Ojectives: Stated clearly for each module of instruction
Prerequisites: None mentioned
Package Description: A notebook containing supplementary information, overhead transparencies for teaching machine operation, and directions for use of each module. System is used primarily for CAI.

RATINGS: 0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree
Content: 3.3 Accuracy: 3 Educational Value: 4 Freedom from biases: 3

Instructional Characteristics: 3.4

Purpose defined:	4	Purpose achieved:	3	Objectives clear:	3	Presented clearly:	4
Response prompts:	3	Outputs make sense:	3	Diff level approp:	3	Learning strategy:	4
Content motivate:	3	Creativity stimul:	3	Feedback helpful:	3	Learner control:	3
Relate prev exp:	4	Generalizability:	4	Use outside mater:	3	Interact varied:	3
Sequence topics:	3	Adequate review:	3	Appropriate use of graphics/color:	3		

Technical Characteristics: 3.0

User operate OK: 3	Teacher can use: 2	Comp use appropr: 3	Reliability: 3
Errors trapped: 3	Effective display: 3	Displ easily read:3	Give good hd copy:3
Pauses appropr: 3	Responses counted: 3	Modification easy:3	Supp mater good: 3

User support materials comprehensive: 3

Overall Rating: 3.3
Purpose fulfilled: 3 Instructional characteristics: 4 Tech characteristics: 3

Potential uses in classroom: Great potential, but teacher would have to choose and select what material to omit if one did not want to spend 4 to 6 weeks on the package.

Time needed to complete package: Four to six weeks

RECOMMENDATIONS: Package is recommended as is.

Major Strengths: Documentation and supplementary materials; one of the few of its kind, i.e., fills a need; well researched; content is accurate.

Major Weaknesses: Could not be used easily by an individual student.

Comments: The documentation's introductory section on microcomputer operation is nice.

EDUCATIONAL SOFTWARE EVALUATION

TITLE: PeachText 5000 **VER:** 2.1 **COPYRIGHT:** Yes **DATE:** 1983
VENDOR: Peachtree Software, Inc. **CONTACT:** Management Science America, Inc.
ADDRESS: 3445 Peachtree Rd., N.E., 8th Floor
CITY: Atlanta **ST:** GA **ZIP:** 30326 **PH:** 1-800-554-8900

PKG TYPE: Integrated Cluster **VOC AREA:** Bus. Educ. **SUBJ AREA:** Word Process.
Accounting
Report
Writing
PURPOSE: Administrative, Data Management, CAI.
PROGRAM IS SUPPOSED TO: This cluster of programs includes a comprehensive word processor, an electronic spreadsheet, a proofreader, a report generator, a Thesaurus, and a set of tutorial lessons. The cluster forms the core of programs most serious computer users in business must learn to use. An on-line set of "Help Screens" is available at any time during programs' operation.

MODE: Tutorial, problem solving, computer as a tool, simulation
PRICE 1ST COPY: $395 **MULT COPIES:** Not needed **UPDATES:** Extra cost
BACKUP OK: Yes **PROGRAM LOCKED:** Assembled code **FIELD TEST AVAILABLE:** No
INSTALLATION ASSISTANCE: Full instructions included **STAFF TRAINING:** CAI only
COMPUTER USED: Most IBM-PC compatibles and many 8-bit Z-80 (CP/M) computers.
HARDWARE REQ'D: IBM compatibles need 128K memory with 2-DS/DD disk drives with 80-column monitor. CP/M 8-bit computers need 64K with 2-SS/DD disk drives. A compatible printer, preferably a word-processing type.
SOFTWARE SUPPORT REQ'D: PC-DOS or MS-DOS 1.1 or later versions. CP/M 2.0 or higher.

DOCUMENTATION: P=In program, S=In suppl. materials, NI=Not incl., NA=Not appl.

Grade/Ability:	NA	Instr Obj:	NA	Prereq skill/act:	NA	Samp progr output:	S
Operating Inst:	P-S	Pretest:	NA	Posttest:	NA	Teacher inform:	S
Resource/ref info:	S	Student instr:	P-S	Student wkshts:	S	Text correlation:	NA
Followup activity:	NA	Program listing:	NA	Index:	S	Table of content:	S
Purpose of pkg:	S	System overview:	S	Intended aud:	S	System capacity:	S
Hardware info:	S	Help section:	P-S	Sample screen:	P-S	Sample report:	S
Sample graphs:	NA	Sample plots:	NA	Interface instr:	S		

DOCUMENTATION RATING: 4.0 (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly Agree)
Readability: 4 Organization: 4 Examples relevant: 4 Provided assist: 4
Consistent in format and terminology: 4

COURSEWARE EVALUATION:
Intended audience: Secondary, Post-secondary, Business
Objectives: To lead the user through samples of how each part of the package is supposed to work, including statements and commands based on useful examples.
Prerequisites: Knowledge of keyboarding, preferably touch typing before use of the word-processing program PeachText, some basic understanding of elementary accounting for use of the PeachCalc in financial planning.

Package Description: An integrated cluster of programs, including a CAI step-by-step tutorial for training users. Cluster includes training for a word processor, a spelling checker, an accounting-type electronic spreadsheet, and a

List Manager. A excellent separate four-disk CAI training system from American Training Institute is included as a supplement to tutor the user through each part of the PeacnText 5000 system. The ATI package should be used prior to the actual hands-on PeachText tutorials.

RATINGS: (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree)
Content: 4 Accuracy: 4 Educational Value: 4 Freedom from biases: 4

Instructional Characteristics: 3.7

Purpose defined: 4	Purpose achieved: 3	Objectives clear: 4	Presented clearly:4
Response prompts:4	Outputs make sense:4	Diff level approp:4	Learning strategy:4
Content motivate:3	Creativity stimula:4	Feedback helpful: 3	Learner control: 4
Relate prev exp: 4	Generalizability: 3	Use outside mater:0	Interact varied: 3
Sequence topics: 4	Adequate review: 4	Appropriate use of graphics/color: 0	

TECHNICAL CHARACTERISTICS: 4.0

User operate OK: 4	Teacher can use: 4	Comp use appropr: 4	Reliability: 4
Errors trapped: 4	Effective display: 4	Displ easily read:4	Give good hd copy:4
Pauses appropr: 4	Responses counted: 0	Modified easily: 0	Supp mater good: 4
User support materials comprehensive: 4			

ADMINSTRATIVE, CMI, OR DATA MANAGEMENT PROGRAMS:
Intended audience: Secondary or later, excellent for individuals or small businesses needing productivity system for word processing, financial planning, and information handling.

RATINGS: (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree)
Inputs and Operations: 3.3

Pkg menu driven: 4	Runs at good speed: 4	Informs on errors and restart: 3
Easy prog restart: 3	Data prompts OK: 3	Users can define code options: 3
Data entry consist: 3	Editing opts adeq: 4	Permit entry of good data only: 0
Opt user def fields:3	Input fld size descr:3	Data fld size OK for school use:3
Sys & rec,capac OK: 3	Rec retrieve by #: 3	Rec easy to retrieve by content:3
Find # of recs used:3	Adequate file maint: 3	Warns when capacity is near: 4
Can copy data files:4		

Outputs: 3.5

Disply easy to read:4	Approp for purpose: 3	Option for screen or hard copy: 4
Reports easy to use:3		

Overall Rating: 3.6

Easy install & use: 4	Advant over manual meth: 4	Flexible for local needs:4
Data generated by progr accessible by other programs: 3	Program is reliable: 3	Does what claims to do: 4
	Confidential data protected:0	Loss of data protected: 3

General Observations:
Purpose: PeachText 5000 is a comprehensive system for processing information that is designed around the word processing system. It is designed to interface with self generated sequential files or BASIC routines. It can be programmed with logic statements similar to BASIC statements. The system includes a powerful electronic spreadsheet for budget manipulation, and other utility programs for creation, editing, formating, and printing the resulting documents or reports with ease and considerable sophistication.

System capacities: The capacity of the system is limited only by the amount of random-access memory and disk space available. Documents generated can be printed in the background mode while the computer operator is generating new material using another program in the cluster.

Possible outputs: Either screen or printer output for documents, letters, form letters, mailing labels, data files, merge mailings, budgets, themes, manuscripts, and most personal or business financial reports or analyses.

Major components:
PeachText word processor, with sample lessons, Disk 1; Random House Electronic Thesaurus, Disk 2; Spelling Proofreader, Disk 3; PeachCalc, Disk 4; List Manager, Disk 5; Configurator (& List Manager Sample Lessons), Disk 6; ATI training package; for PeachText, PeachCalc, and List Manager; Reference Guide, Lesson Plan manual, Quick start instructions, Word Processor Reference Card, and PeachCalc Reference Card.

Other possible applications: Limited only by the user's imagination.

RATING SUMMARY: 3.75
Purpose fulfilled: 4 Instructional characteristics: 4 Tech characteristics: 4
Management characteristics: 3

Potential uses in classroom: It has great potential as an easy to learn system with a substantial use in the business world. The instructional package has been used with considerable success in training. Since the system is built around a word processing system, its use in data file generation makes very rapid searching and retrieval of information possible with a minimum of effort. Both the instructional packages and the actual operational cluster are good examples of what could be done with individualized instruction. Some motivation to read the manuals in order to learn the system would be necessary.

Time needed to complete package: A full semester at an hour a day could be used to become proficient with all phases of the PeachText program. A similar time could be spent on the PeachCalc program. A few hours spent with the training package could make the basic system functional in each case. Actual practice and reference to the included materials would be needed to develop proficiency.

RECOMMENDATIONS: Package is recommended for use.
Major Strengths: Low cost (available at considerable discount), ease of learning and use, adequate and appropriate documentation, versatility.

Major Weaknesses: None, although one needs to become familiar with the "Helps" screens or documentation to become proficient with the package's power.

Comments: The ATI training program makes this package an excellent buy for teaching students or users how to use this comprehensive productivity system of information handling for word and data processing. Its command structure is logical and straightforward. Its outputs include proportional printing and margin justification and many other outstanding features at a low cost.

EDUCATIONAL SOFTWARE EVALUATION

TITLE: The Store Manager **VER:** 5.6 **COPYRIGHT:** Yes **DATE:** Aug. 1978
VENDOR: High Technology Software, Inc.
ADDRESS: P.O. Box 60406
CITY: Oklahoma City **ST:** OK **ZIP:** 73146 **PH:** 405-524-5249
PKG TYPE: Integrated cluster **VOC AREA:** Bus./D.E. **SUBJ AREA:** Mark./Acctng Retailing
PURPOSE: Store management program, using data management techniques.
PROGRAM IS SUPPOSED TO: Manage all retailing data, inventory control, mailing lists, record storage, and invoicing. Excellent for instructional simulation.
MODE: Tutorial, computer as a tool, record keeping
PRICE 1ST COPY: $295 Educational discounts may be requested.
BACKUP OK: No **PROGRAM LOCKED:** Yes **FIELD TEST AVAILABLE:** No
INSTALLATION ASSISTANCE: YES, by phone, 90 days **STAFF TRAINING:** No
COMPUTER USED: AppleII 48K memory
HARDWARE REQ'D: 2-3 SS/SD 5-1/4" disk drives, 40 Col monitor, printer
SOFTWARE SUPPORT REQ'D: APPLEDOS 3.3 with BASIC

DOCUMENTATION: P=In program, S=In suppl. materials, NI=Not incl., NA=Not appl.

Grade/Ability:	S	Instr Obj:	NI	Prereq skill/act:	NI	Samp progr output:	S
Operating Inst:	S	Pretest:	NI	Posttest:	NI	Teacher inform:	S
Resource/ref info:	S	Student instr:	S	Student wkshts:	NA	Text correlation:	NI
Followup activity:	NA	Program listing:	NI	Index:	S	Table of content:	S
Purpose of pkg:	S	System overview:	S	Intended aud:	NI	System capacity:	S
Hardware info:	S	Help section:	P-S	Sample screen:	S	Sample report:	P-S
Sample graphs:	NA	Sample plots:	NA	Interface instr:	S		

DOCUMENTATION RATING: 4.0 (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly Agree)
Readability: 4 Organization: 4 Examples relevant: 4 Provided assist: 4
Consistent in format and terminology: 4

COURSEWARE EVALUATION:
Intended audience: Sec./Post Sec. Bus. and Dist. Educ. **Size:** Indiv./small group
Objectives: Inferred in documentation and program (Not specifically for educ.)
Prerequisites: Inferred in program
Package Description: This package is well designed for retailing. Can be used for instruction since the detailed records such as inventory, shipping, resale, etc. are presented in actual applications.

RATINGS: (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree)
Content: 4.0 Accuracy: 4 Educational Value: 4 Freedom from bias: 4

Instructional Characteristics: 3.7

Purpose defined:	3	Purpose achieved:	4	Objectives clear:	4	Presented clearly:	4
Response prompts:	4	Outputs make sense:	4	Diff level approp:	3	Learning strategy:	0
Content motivate:	4	Creativity stimula:	4	Feedback helpful:	0	Learner control:	0
Relate prev exp:	0	Generalizability:	3	Use outside mater:	0	Interact varied:	0
Sequence topics:	3	Adequate review:	0	Appropriate use of graphics/color:	3		

TECHNICAL CHARACTERISTICS:

User operate OK:	3	Teacher can use:	3	Comp use appropr:	4	Reliability:	4
Errors trapped:	4	Effective display:	2	Displ easily read:	4	Give good hd copy:	4
Pauses appropr:	4	Responses counted:	0	Modified easily:	0	Supp mater good:	4
User support materials comprehensive:	4						

ADMINSTRATIVE, CMI, OR DATA MANAGEMENT PROGRAMS:
Intended audience: Secondary or post-secondary business/distiibutive education

RATINGS: (0=NA, 1=Disagree, 2=Neutral, 3=Agree, 4=Strongly agree)
Inputs and Operations: 4.0

Pkg menu driven:	4	Runs at good speed:	4	Informs on errors and restart:	4
Easy prog restart:	4	Data prompts OK:	4	Users can define code options:	0
Data entry consist:	4	Editing opts adeq:	4	Permit entry of good data only:	4
Opt user def fields:	0	Input fld size descr:	4	Data fld size OK for school use:	4
Sys & rec capac OK:	4	Rec retrieve by #:	4	Rec easy to retrieve by content:	4
Find # of recs used:	4	Adequate file maint:	4	Warns when capacity is near:	0
Can copy data files:	4				

Outputs: 4.0

Disply easy to read:	4	Approp for purpose:	4	Option for screen or hard copy:	4
Reports easy to use:	4				

Overall Rating: 4.0

Easy install & use:	4	Advant over manual meth:	4	Flexible for local needs:	0
Data generated by progr accessible by other programs:	0	Program is reliable:	4	Does what claims to do:	4
		Confidential data protected:	4	Loss of data protected:	4

General Observations:
Purpose: An excellent package to train students about the necessary administrative and record-keeping needs of a retail store.
System capacities: Provides for order processing, inventory control, purchasing and management reports.
Possible outputs: Inventory value, turnover, and checklist reports, item movement reports, customer lists, quotations, reciepts, packing lists, order forms, backorder reports.
Major components: Store Manager, Customer Data File, and Inventory Data File
Other possible applications: Accounting tutor

RATING SUMMARY: 3.7
Purpose fulfilled: 4 Instructional characteristics: 3 Tech characteristics: 4
Potential uses in classroom: To familiarize students with a comprehensive system of using invoices, quotations, reciepts, packing lists, inventory control, and updating procedures.
Time needed to complete package: One week at one hour per day.

RECOMMENDATIONS: Package is highly recommended as a real world application.
Major Strengths: Comprehensive in content and easy to use.
Major Weaknesses: Highly structured with no means of modification.

Comments: Should provide assistance in instruction by offering a structured thought process for handling daily retailing transactions.

REFERENCES

Anderson, R. C., Kulhavy, R. W., & Andre, T. (1971). Feedback procedures in programmed instructions. Journal of Educational Psychology. 62, 148.

Anderson, R. C., Kulhavy, R. W., & Andre, T. (1972). Conditions under which feedback facilitates learning from programmed lessons. Journal of Educational Psychology 63, 186.

Barnes, K. (1984). Selecting 'Good' stuff in computer training in a 2-sided job. PC Week, 1(10), 23.

Boas, E., Frantz, M., & Matthews, J. (1978). Computer-managed instruction for staff development in a community college system. Proceedings of association for Educational Data Systems, 16th annual convention. Atlanta, GA.

Bork, A. (1978). Machines for computer-assisted learning. Educational Technology, 18(4), 17-22.

Bruner, J. S. (1966). Toward a theory of instruction. Cambridge, MA: Harvard University Press.

Charles, C. M. (1976). Individualizing instruction. St. Louis: C. V. Mosby.

Chase, S. A., Gordon, R., & Makin, R. C. (1984). A system for evaluating microcomputer courseware for vocational and technical education. Columbus, OH: The National Center for Research in Vocational Education.

Cohen, V. L. (1982, March). Evaluating instructional software for the microcomputer. Paper presented at the annual meeting of the American Educational Research Association, New York. (ERIC Document Reproduction Service No. ED 216 704).

Crawford, S. (1981). A standard's guide for the authoring of instructional software. Reference manual volume III. Victoria, British Columbia: Joint Educational Management (JEM) Research. (ERIC Document Reproduction Service No. ED 211 062.)

DiVesta, F. J. Trait-treatment interactions, cognitive processes, and research on communications media. AV Communications Review. 23, 185-96.

Ellington, H., Addinall, E., & Percival, F. (1981). Games and simulations in science education. London: Kogan Page.

Frankel, P. & Grass, A. (1983). The software sifter: An Intelligent shopper's guide to buying computer software. New York: Macmillan.

Frantz, N., Matthews, J. I., & Boas, E. (1979). Using computer-managed instruction for staff development at a technical and community college. T. H. E. JOURNAL. March, 30-31.

Grunden, H. U. (1969). Response mode and information about correct answers in programmed instruction. In A. P. Mann and C. K. Burunstrom (Eds.). Aspects of educational technology III. London: Pittman.

Harper, D. O., & Stewart, J. H. (1983). RUN: Computer education. Monterey, CA: Brooks/Cole.

Hartley, J. R., & Lovell, K. (1977). The psychological principles underly ing the design of computer-based instructional systems. In D. F. Walker & R. D. Hess (Eds.). (1984). Instructional software: Principles and perspectives for design and use, (pp. 38-56). Belmont, CA: Wadsworth.

Hofmeister, A. (1984). Microcomputer applications in the classroom. New York: Holt, Rinehart and Winston.

Malone, T. W. (1981). Toward a theory of intrinsically motivating instruction. Cognitive Science, 4, 333-69.

Maslov, A. H. (1954). Motivation and personality. New York: Harper and Brothers.

Matthews, J. I. (1978). Microcomputer vs. minicomputer for educational computing. Educational Technology. 18(11), 19-22.

Northwest Regional Educational Laboratory. (1981). Evaluator's guide for microcomputer-based instructional packages. Portland, OR. (ERIC Document Reproduction Service No. ED 206 330).

Ornstein, R. (1977). The psychology of consciousness (2nd ed.). New York: Harcourt Brace Jovanovich.

Papert, S. (1980). Computers and computer cultures. In D. O. Harper & J. H. Stewart (Eds.). (1983). Teaching and learning theory (Chapter 1), Run: Computer education (pp. 3-10). Monterey, CA: Brooks/Cole.

Piaget, J. (1971). Science of education and the psychology of the child. New York: Viking Press.

Post, P. E., & Sarapin, M. I. (1983, December). Writing and evaluating educational software: Some key elements. Paper presented at the 1983 annual meeting of the American Vocational Association, Anaheim, CA.

Pucel, D. J., & Knaak, W. C. (1975). Individualized vocational and technical instruction. Columbus, OH: Charles E. Merrill.

Rowe, N. C. (1981). Some rules for good simulations. In D. F. Walker & R. D. Hess (Eds.). Instructional software: Principles and perspectives for design and use (pp. 181-86). Belmont, CA: Wadsworth.

Sagan, C. (1977). Dragons of Eden. New York: Random House.

Southwest Educational Development Laboratory. (1983). Evaluation of educational software: A guide to the guides. Austin, TX.

Steffin, S. A. (1983). A suggested model for establishing the validity of computer-assisted instructional materials. Educational Technology, 23,(1), 20-22.

Steinberg, E. R. (1977). Review of student control in computer-assisted instruction. Journal of computer-based instruction, 3 84-90.

Stone, A. (1983). Microcomputer software for adult vocational education: Guidelines for evaluation. Columbus, OH: The National Center for Research in Vocational Education.

Tennyson, R. D., & Buttrey, T. (1980). Advisement and management strateg ies as design variables in computer-assisted instruction. In D. F. Walker & R. D. Hess (Eds.). (1984). Instructional software: Principles and perspectives for design and use (pp. 158-65). Belmont, CA: Wadsworth.

Walker, D. F. & Hess, R. D. (Eds.). (1984). Instructional software: Principles and perspectives for design and use. Belmont, CA: Wadsworth.

Wang, R. R. (1982). The design and development of a microcomputer-managed delivery system for individualized instruction in vocational education. (Doctoral dissertation, The University of Tennessee, Knoxville, 1982).

Weaver, D. & Holznagel, D. (1984, April). An analysis of available course ware. Reports to Decision Makers. Portland, OR: Northwest Regional Educational Laboratory.

HIGH TECHNOLOGY EDUCATION: A PROGRAM OF WORK

The following publications have been developed by the Office for Research in High Technology Education for the U.S. Department of Education's Office of Vocational and Adult Education:

At Home in the Office:

- At Home in the Office: A Guide for the Home Worker

COMTASK:

- Procedures for Conducting a Job Analysis: A Manual for the COMTASK Database
- COMTASK User's Guide

State-of-the-Art Papers:

- The Changing Business Environment: Implications for Vocational Curricula
- Computer Literacy in Vocational Education: Perspectives and Directions
- Computer Software for Vocational Education: Development and Evaluation
- Educating for the Future: The Effects of Some Recent Legislation on Secondary Vocational Education
- The Electronic Cottage
- High Technology in Rural Settings
- (Re)Training Adults for New Office and Business Technologies
- Robots, Jobs, and Education
- Work in a World of High Technology: Problems and Prospects for Disadvantaged Workers